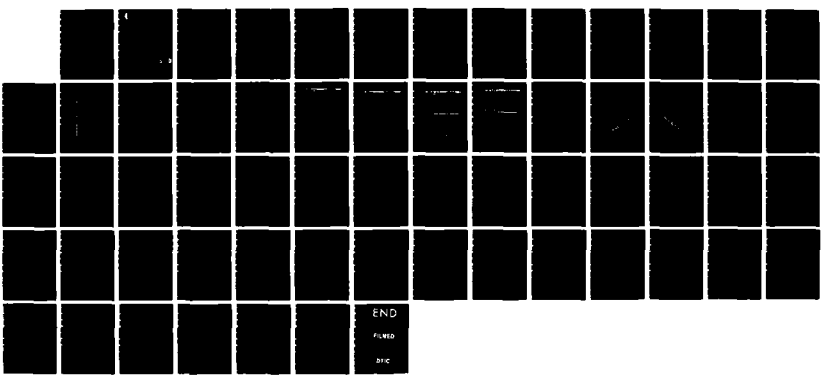
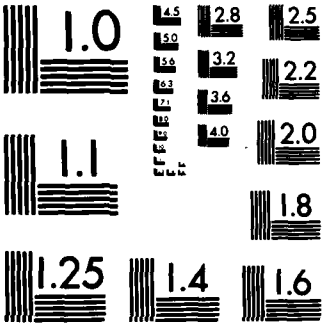
AD-A148 559 AMERICAN CANADIAN AUSTRALIAN BRITISH URBAN GAME (ACABUG) AN OVERVIEW(U) ARMY TRADOC SYSTEMS ANALYSIS ACTIVITY WHITE SANDS MISSILE RANGE NM 1/1

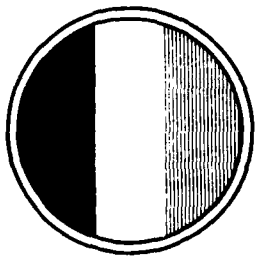
UNCLASSIFIED D A HARTLEY ET AL. OCT 83 TRASANA-TR-31-83 F/G 9/2 NL



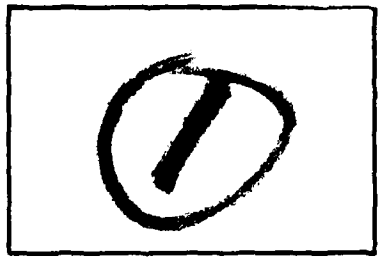


MICROCOPY RESOLUTION TEST CHART

)ARDS-1963-A



ACN 67504



TRASANA-TR-31–83

AMERICAN CANADIAN AUSTRALIAN BRITISH URBAN GAME (ACABUG)

AN OVERVIEW

PREPARED BY

David A. Hartley

Raymond B. Heath

OCTOBER 1983

Approved for Public release;
distribution is unlimited.

DEPARTMENT OF THE ARMY
US ARMY TRADOC SYSTEMS ANALYSIS ACTIVITY
WHITE SANDS MISSILE RANGE
NEW MEXICO 88002

Encl 1

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR-31-83	2. GOVT ACCESSION NO. AD-A148 559	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* American Canadian Australian British Urban Game (ACABUG), An Overview		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dr. David A. Hartley Mr. Raymond B. Heath		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army TRADOC Systems Analysis Activity ATTN: ATOR-TAB White Sands Missile Range, NM 88002		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Training and Doctrine Command ATTN: ATCD-AC Fort Monroe, VA 23651		12. REPORT DATE October 1983 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE UNCLASSIFIED

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
PASCAL, Wargame, ABCA, Simulation, Computer-Assisted, MOUT, Urban

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*
The four ABCA Nations (America, Britain, Canada, Australia) have jointly developed a computer assisted MOUT (Military Operations in Urban Terrain) wargame named ACABUG (American Canadian Australian British Urban Game). Although designed primarily as a MOUT model capable of representing a defensive reinforced infantry company against an appropriate threat force, ACABUG has of necessity sufficient generality that it can handle a reinforced infantry company or an armored battalion in either a rural or an urban (OVER)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

setting. This report describes in general terms the structure and features of the ACABUG software, which has been implemented on a PERQ microcomputer in the PASCAL programing language.

TRASANA TECHNICAL REPORT 31-83
AMERICAN CANADIAN AUSTRALIAN BRITISH URBAN GAME
(ACABUG)
AN OVERVIEW

TABLE OF CONTENTS

	Page
ABSTRACT	ix
1. BACKGROUND TO ACABUG	1
2. AIM	2
3. OVERVIEW OF THE ACABUG SOFTWARE	2
4. INPUT OF COMMANDS TO THE COMPUTER	8
5. OUTPUT	14
6. THE FUTURE EVENT HANDLER	18
7. THE SCHEDULING AND ASSESSMENT ALGORITHMS	20
8. NON-FIRING TARGET ACQUISITION	22
9. FLASH DETECTION	23
10. COMMUNICATONS	25
11. INDIRECT FIRE	25
12. DIRECT FIRE	27
13. SUPPRESSION	30
14. MOUNTING AND DISMOUNTING PLATFORMS	31
15. MOVEMENT (AIR AND GROUND)	33
16. BUILDING CLEARANCE	36
17. THE POST PROCESSOR	38
18. BATTLE LOG	41
19. LINE-OF-SIGHT	42
20. GAME PARAMETER UPDATE	44
21. LIST INFORMATION	44
22. STATUS UPDATE	46

TABLE OF CONTENTS CONTINUED

	Page
APPENDIX A. SUPPRESSION ALGORITHM	A-1
REFERENCES	xi
DISTRIBUTION	xiii

TRASANA TECHNICAL REPORT 31-83
AMERICAN CANADIAN AUSTRALIAN BRITISH URBAN GAME
(ACABUG)
AN OVERVIEW

LIST OF FIGURES

		Page
FIGURE 1.	ACABUG Procedural Structure	3
FIGURE 2.	ACABUG Software Structure	5
FIGURE 3.	ACABUG Data Flow	6
FIGURE 4.	Exercise Run Logic Flow	7
FIGURE 5.	Main Level Command Window	10
FIGURE 6.	Run-Time Menu Window	11
FIGURE 7.	Direct Fire Order Window	12
FIGURE 8.	List Information and Example Information Window	13
FIGURE 9.	Optical Mark Reader Form For Indirect Fire	15
FIGURE 10.	Optical Mark Reader Form For Ground Movement	16
FIGURE 11.	Example ACABUG Killer Victim Scoreboard	42

ABSTRACT

The four ABCA Nations (America, Britain, Canada, Australia) have jointly developed a computer assisted MOUT (Military Operations in Urban Terrain) war game named ACABUG (American Canadian Australian British Urban Game). Although designed primarily as a MOUT model capable of representing a defensive reinforced infantry company against an appropriate threat force, ACABUG has of necessity sufficient generality that it can handle a reinforced infantry company or an armored battalion in either a rural or an urban setting. This report describes in general terms the structure and features of the ACABUG software, which has been implemented on a PERQ microcomputer in the PASCAL programming language.

TRASANA TECHNICAL REPORT 31-83
AMERICAN CANADIAN AUSTRALIAN BRITISH URBAN GAME (ACABUG) AN OVERVIEW

1. BACKGROUND TO ACABUG

a. The four ABCA Nations (America, Britain, Canada, Australia) have entered a program to jointly develop one or more models pertaining to MOUT (Military Operations in Urban Terrain). The aim of the program is to improve the capabilities of the four nations for undertaking analytic studies of MOUT related issues, whether they be weapons design, force mix or tactical problems.

b. The program is being managed by the SWP/MOUT (Special Working Party for MOUT) under the general auspicies of the QWG/AOR (Quadripartite Working Group for Army Operations Research). In its report to the 13th meeting of the QWG/AOR in April 1982 the SWP/MOUT recommended a phased development of MOUT models, starting with a computer assisted MOUT war game to be named ACABUG (American Canadian Australian British Urban Game). Reference 1 contains further information concerning the background to this recommendation.

c. ACABUG is a computer assisted war game using three dimensional terrain boards for the determination of line of sight. Players move and deploy markers on the terrain board to establish line of sight, detection and engagement opportunities. Any engagements are initiated by players by means of an input order to the computer. The role of the computer is mainly to assess the results of engagements and feed information back to the players. The computer also performs a bookkeeping role. Reference 2 contains a more detailed account of general war gaming techniques.

d. ACABUG is based very much on concepts taken from the US Army's BATTLE (Battalion Analyzer and Tactical Trainer for Local Engagements) War Game (Reference 3) and the UK RARDE Battle Group War Game. Building on these two war games as a basic source of ideas a Concept Design Report (Reference 4) was produced in October 1982. As expected, some details have changed during the implementation phase as experience has been gained and ideas refined, rendering the Concept Design Report occasionally out of date on points of detail. However, the implementation does follow closely the general guidelines laid down in that report, and extensive reference to that report is made here.

e. ACABUG has been implemented on a PERQ microcomputer produced by Three Rivers Computer Corporation, the software being written in PASCAL. The computer is a 16-bit microcomputer with one megabyte of random access memory and a 24 megabyte, non-removable Winchester disk. A high-resolution, bit mapped graphics display is provided as standard, which features a 768 x 1024 pixel display (black on white) on an 8.5 x 11 inch screen. A graphics tablet and "pointing" device, which controls a screen cursor, are also provided as standard. These facilities, together with the PERQ's multiple window capability

enable advanced graphical input to be used rather than the more conventional keyboard input. Extensive use has been made of this graphical input capability to enhance the user/computer interface.

f. An interim software manual (Reference 5) is nearing completion which describes in some detail the structure of the ACABUG software, the data base interface procedures, the future event handler and the various assessment algorithms. A complete set of ACABUG documentation is under preparation and will be published in due course.

2. AIM. The aim of this report is to outline the general structure and features of the ACABUG software as they currently exist.

3. OVERVIEW OF THE ACABUG SOFTWARE

a. The ACABUG system of programs consists of five major sub-systems:

(1) END-DAY

(2) START-DAY

(3) DATA BASE EDITOR

(4) EXERCISE RUN

(5) POST-PROCESSOR

These are described in more detail in the following paragraphs, and are shown schematically in Figure 1.

b. End-Day. The END-DAY program:

(1) Searches for all data files with a specified security classification, e.g., CONFIDENTIAL or higher.

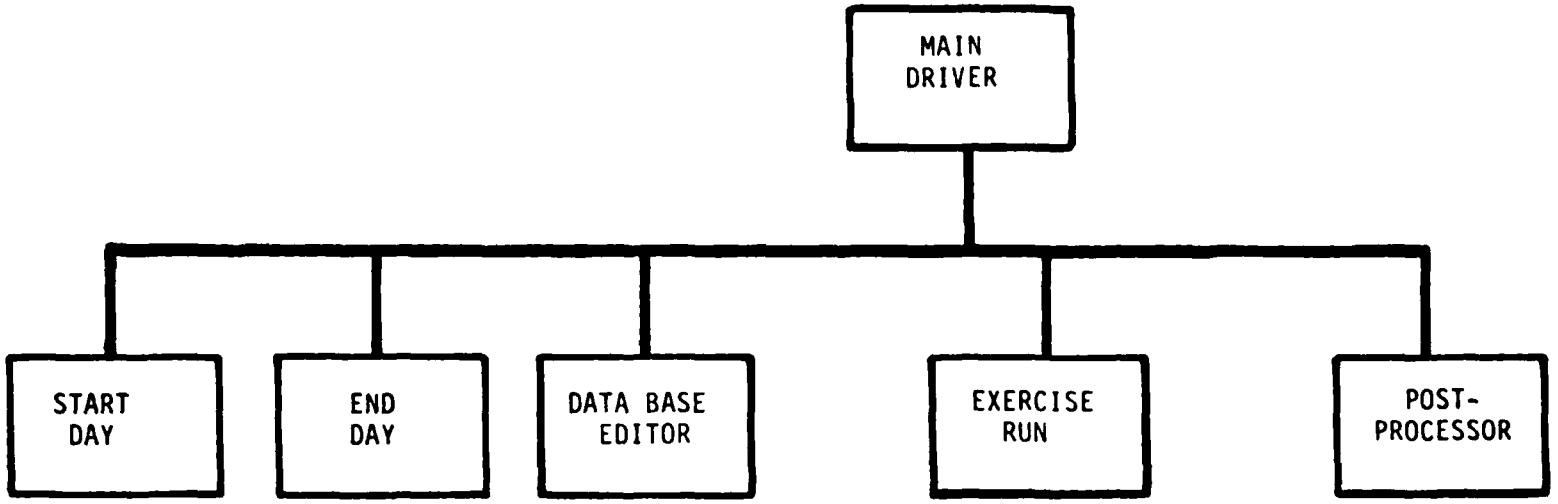
(2) Copies each classified file to floppy disk.

(3) Verifies that it has copied correctly.

(4) Erases the classified files from the Winchester disk.

c. Start-Day. The START-DAY program is the reverse of the END-DAY program. It takes a floppy disk and reads in classified data files that have previously been output to that device using END-DAY. These files are copied onto the Winchester disk, and the security classification of each file is properly recorded.

d. Data Base Editor. This covers both the static (weapons effects and terrain representation) and the dynamic (force organization and unit status) data bases. It contains procedures to:



INITIALIZE DISK
READ-IN CLASSIFIED FILES
INITIALIZE MODEL PARAMETERS
HOT START/ COLD START OPTION

SAVE CLASSIFIED FILES
ERASE CLASSIFIED FILES

WEAPON EFFECTS DATA
URBAN TERRAIN DATA
FORCE ORGANIZATION DATA
UNIT STATUS DATA
INITIAL CONDITIONS

AIR MOVEMENT
BATTLE LOG ENTRY
BUILDING CLEARANCE
COMMUNICATION
DIRECT FIRE
GROUND MOVEMENT
INDIRECT FIRE
LOS
MINEFIELD
MOUNT/DISMOUNT
NON-FIRING ACQUISITION
OPTICAL MARK READER INPUT
EVENT LIST MODIFICATION
GAME PARAMETER UPDATE
LIST INFORMATION
STATUS UPDATE
EXECUTE NEXT EVENT
EXECUTE TO GAME TIME
EXECUTE n EVENTS

LIST HISTORY FILE
KILLER/VICTIM SCOREBOARD
OTHER OPTIONS TO BE IMPLEMENTED AS NEEDED

FIGURE 1. ACABUG Procedural Structure

(1) Read in data from the keyboard.

(2) Edit the various data files.

(3) List the various data files.

e. Exercise Run. This actually runs an ACABUG war game and produces a history file of user selected events that take place during the game.

f. Post-Processor. This analyzes the history file produced during an ACABUG game and generates various analyses such as a killer-victim scoreboard.

g. Data Flows. Figures 2 and 3 illustrate how the various program modules interact with the data base. It should be noted that:

(1) A data-base interface module is needed within the software to interface the higher level ACABUG procedures with the physical storage of data files on the disk. However, a user of ACABUG need not be aware of the method of operation, or even existence of this module as he never uses it directly; it is always called indirectly from other programs.

(2) There are data structure modules. These contain the specifications of data types (mainly PASCAL records) and data base access procedures (which in turn use the data base interface).

(3) The Start-Day and End-Day programs read classified files from or write classified files to a removable floppy disk. This enables classified data to be removed from the computer as an aid to data security.

h. Exercise Run Logic Flow.

(1) Figure 4 illustrates in schematic form the overall logic flow during execution of an ACABUG exercise run.

(2) The focus of the diagram is the box labeled "Choose Menu Option." When the program is in this state, a menu of options appears on the computer screen, from which the user (player) is free to choose. If the user wishes to input an order (e.g., initiate a direct fire mission) he chooses the appropriate option and inputs the necessary information (e.g., firing platform, type of ammunition fired, target platform, number of rounds fired). The machine processes this information, updates the future event list as necessary, and returns to the "Choose Menu Option" state. The actual method of entering information is not addressed in this report.

(3) When there are no further orders to be input then one of the "Execute" options is chosen. Game time is then advanced to the next event, and that event is processed. This may in turn involve further changes to the future event list. After processing the event (or events) the program returns once again to the "Choose Menu Option" state. At this stage further orders may be input, or the next event may be processed.

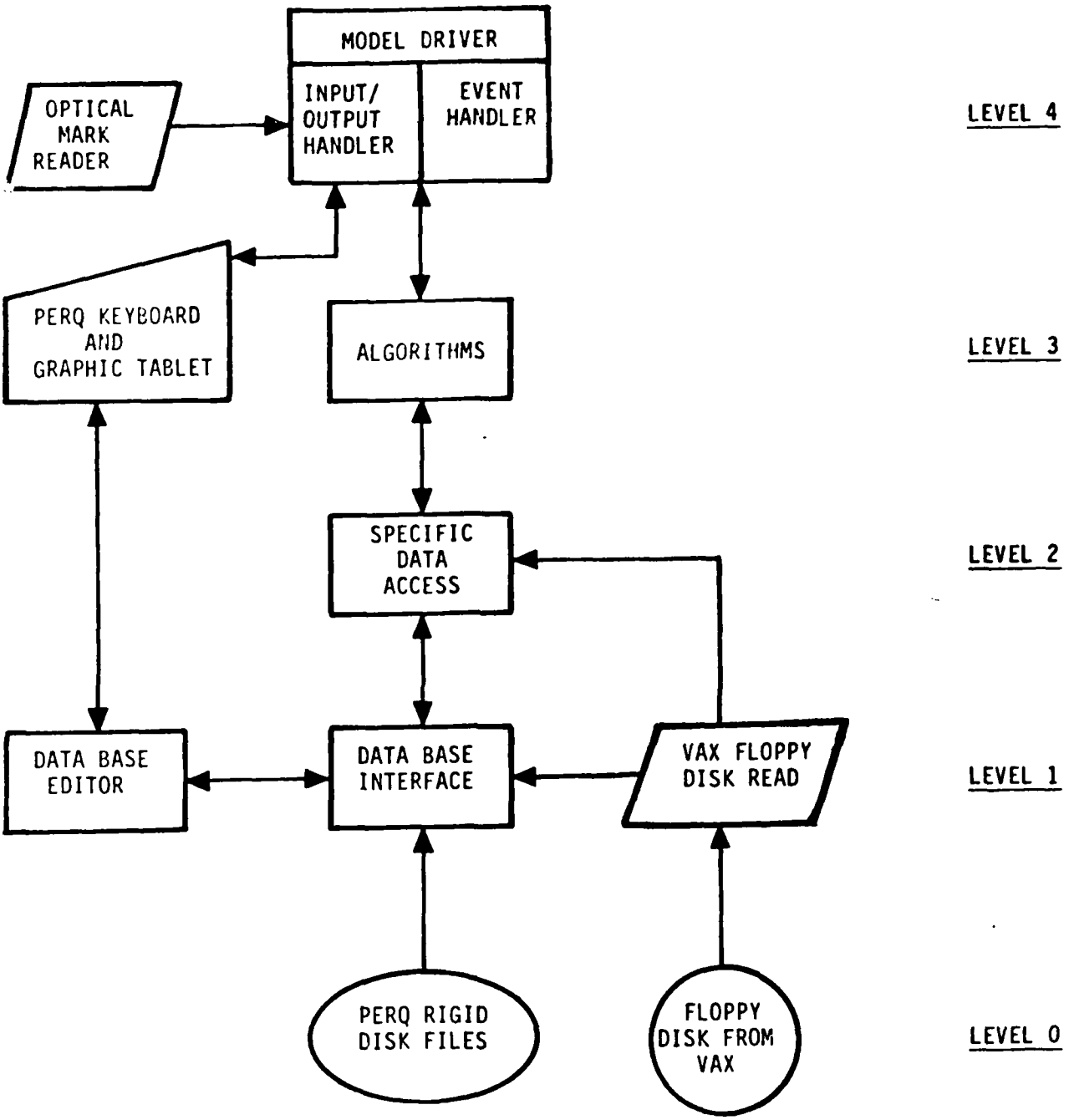


FIGURE 2. ACABUG Software Structure

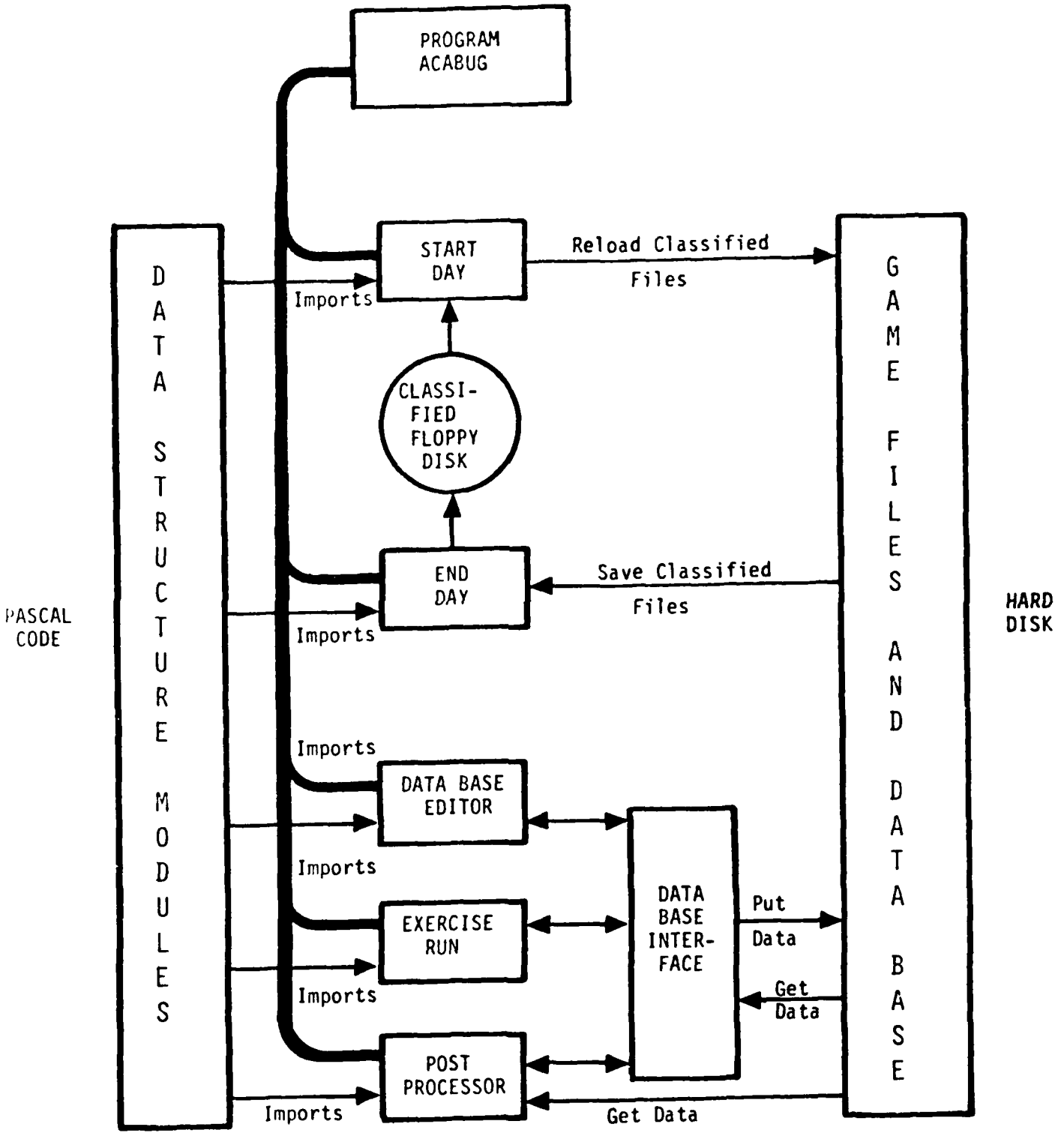


FIGURE 3. ACABUG Data Flow

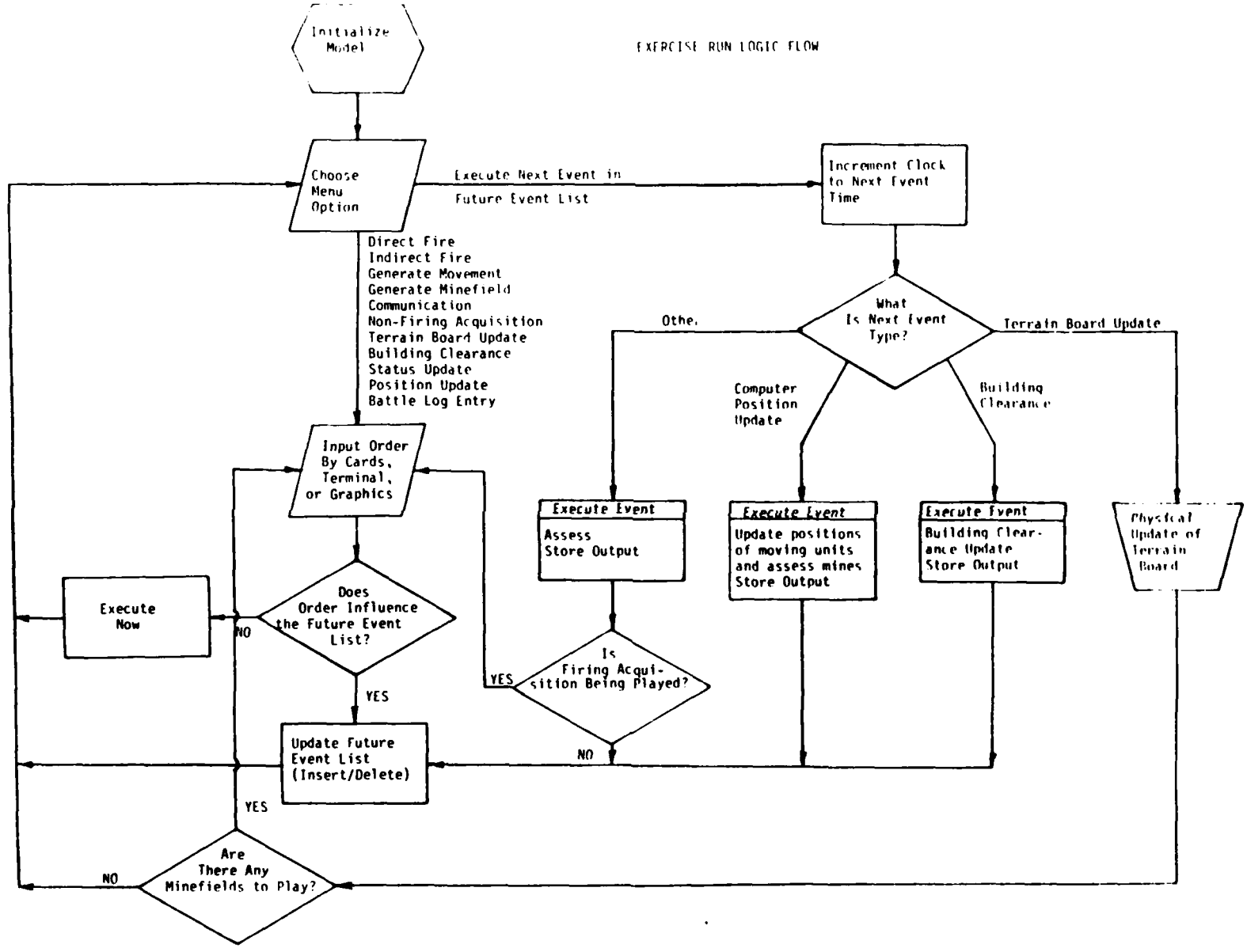


FIGURE 4. Exercise Run Logic Flow

(4) At periodic time intervals it will be necessary to update the physical location of units on the terrain board. In outline the procedure is to move the physical units ahead of time to the positions they will occupy at the time of the next "Physical Update of the Terrain Board" event. This will cause movement orders to be prepared, and target detection and engagement opportunities to be identified, and orders prepared as necessary. All of these orders are input to the computer. As the computer then executes next events, so time is advanced and the events processed. Some of these events will cause unit locations within the computer to be updated in accordance with the movement orders. Eventually, the next "Physical Update of Terrain Board" event will be reached. At this time the computer will have "caught up" with the terrain board locations, and the two sets of locations (physical location on terrain board and location stored in the computer) should agree. Unit locations on the terrain board are now moved ahead of time to their positions at the following "Physical Update of Terrain Board" event, and the loop is repeated.

4. INPUT OF COMMANDS TO THE COMPUTER

a. Since ACABUG is a computer assisted war game all the key decisions are made outside the computer by players and/or controllers. Thus commands must be input to the computer:

(1) As a direct result of orders issued by players, e.g., fire and movement orders.

(2) In order to display information stored in the computer, either for the benefit of players or controllers.

(3) To make modifications to the current force status as a result of assessments made by the controller using supplementary manual rules.

b. Because of the large amount of communication required with the computer, the "computer interface" is an extremely important part of war game design. In particular it is required that:

(1) The preparation of orders for input to the computer should be rapid and the entries unambiguous.

(2) Errors in orders should be trapped as early as possible and in a "user friendly" manner so that the error can quickly be identified and corrected, and the order re-input.

c. It has been attempted to satisfy the above criteria in ACABUG by providing two methods of input, both relatively novel for war gaming use:

(1) Via the PERQ terminal using a "command" package written by New Mexico State University under contract to TRASANA.

(2) Via an optical mark document reader.

d. "Command" Package.

(1) The PERQ computer has as standard high resolution, black-on-white graphics display and a screen cursor controlled by the position of a "puck" or pointing device which is moved over a graphics tablet.

(2) Using the command package, the ACABUG "windowing" has been implemented such that more than one window may be displayed on the screen at the same time. However, the currently active window will overlay all commands in all other windows that may not be executed. Therefore, at all times, only valid commands are available for execution. The currently active window may be changed in two ways:

(a) By placing the screen cursor within any visible window on the screen and then "pushing" one of the buttons on the "puck". This action will immediately transfer program control to that window. This may be done at any time the program is waiting for a "push" (the operator is prompted this status in the main window).

(b) By inputing an order which automatically activates a window. This process is described below.

(3) When orders are input to the computer, the appropriate "window" is called up and the correct entries then selected. The selection is made by moving the screen cursor (using the "puck") to the relevant location, when the selected command goes to inverse video to show it is being selected.

(a) In most cases, when only a small number of alternatives are available, each alternative is explicitely displayed. To make a selection all that is needed is to move the cursor onto the desired selection and press one of the buttons on the "puck."

(b) In a number of cases there are so many alternatives (e.g., the firing platform number for a direct fire order) that the previous approach is not feasible. In these instances the cursor is again moved onto the desired selection, but a numeric entry must then be made via the keyboard.

(4) Whenever possible, defaults have been provided (recognizable by already being in inverse video when the "window" is called up), so as to minimize the number of selections that generally have to be made. Figures 5-8 provide a few examples of some of the available "windows."

(5) In the lower left hand corner of each window, where orders are input, is the Accept Inputs command list. The default is No; the other commands are described below.

(a) Yes. A "push" on this command means that the user considers the order complete and wants to try to input the order. However, each window has extensive error checking for input order correctness. If the order has errors, the error messages are printed in the lower center portion of the window.

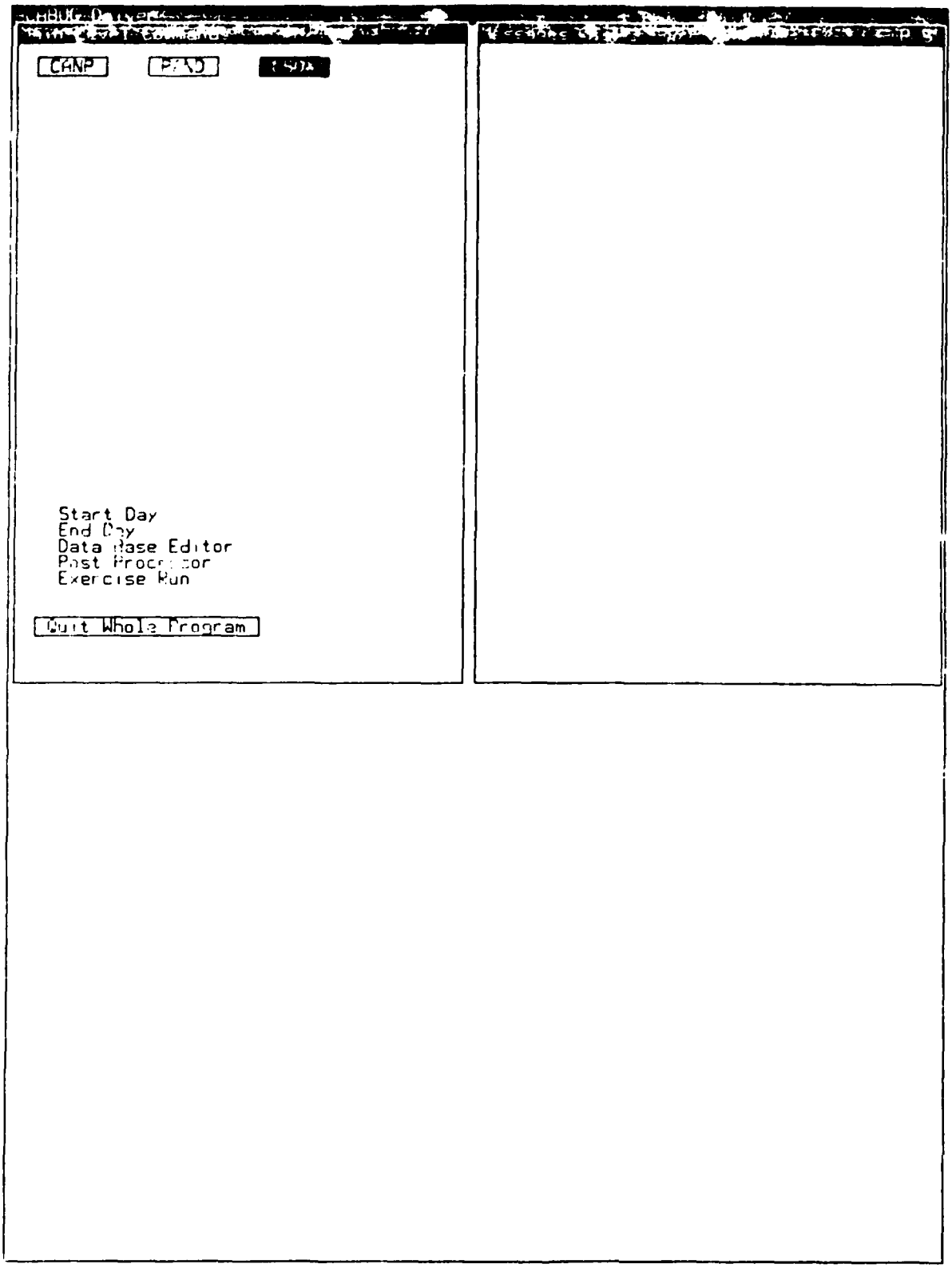


FIGURE 5. Main Level Command Window

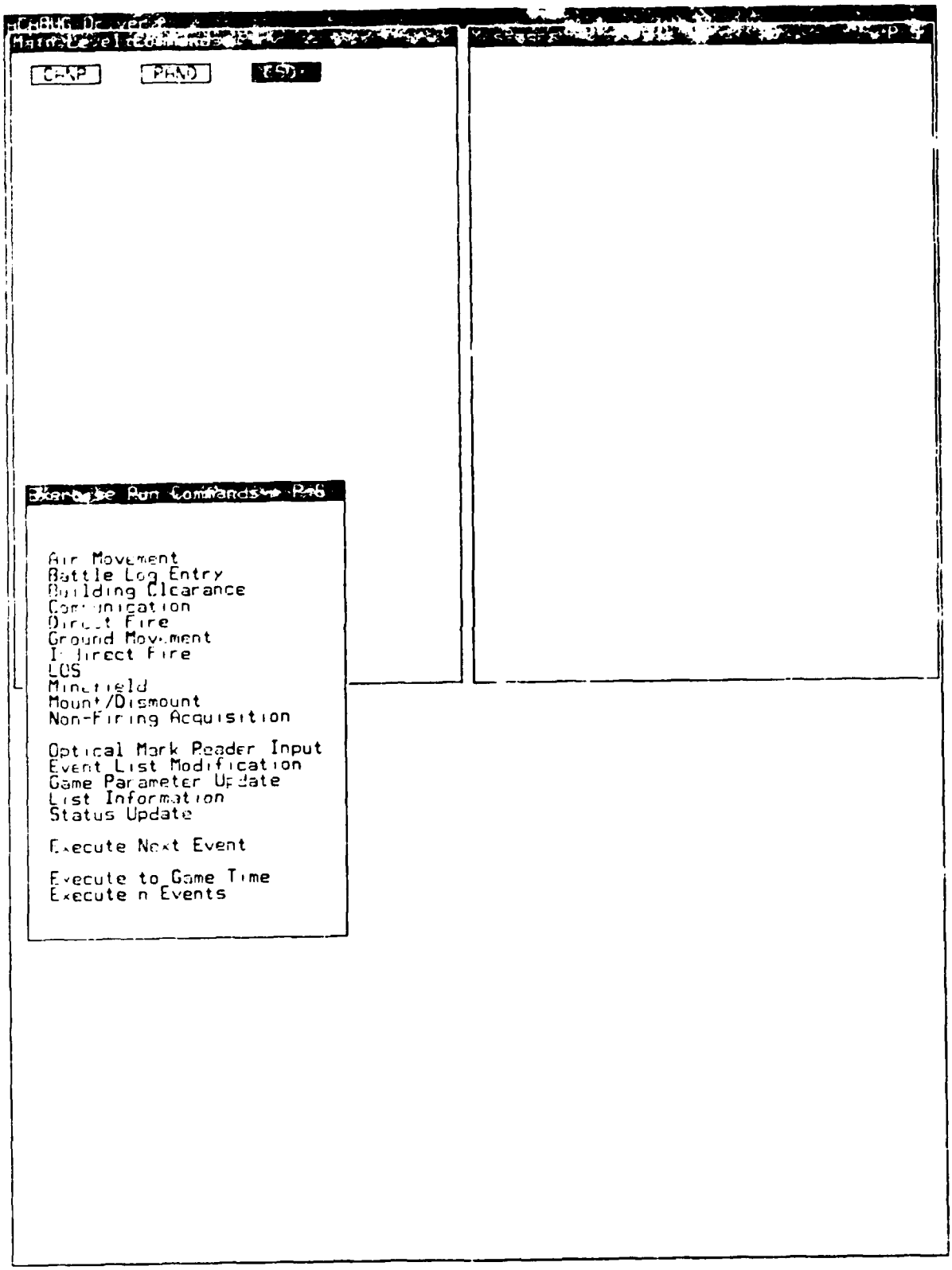


FIGURE 6. Run-Time Menu Window

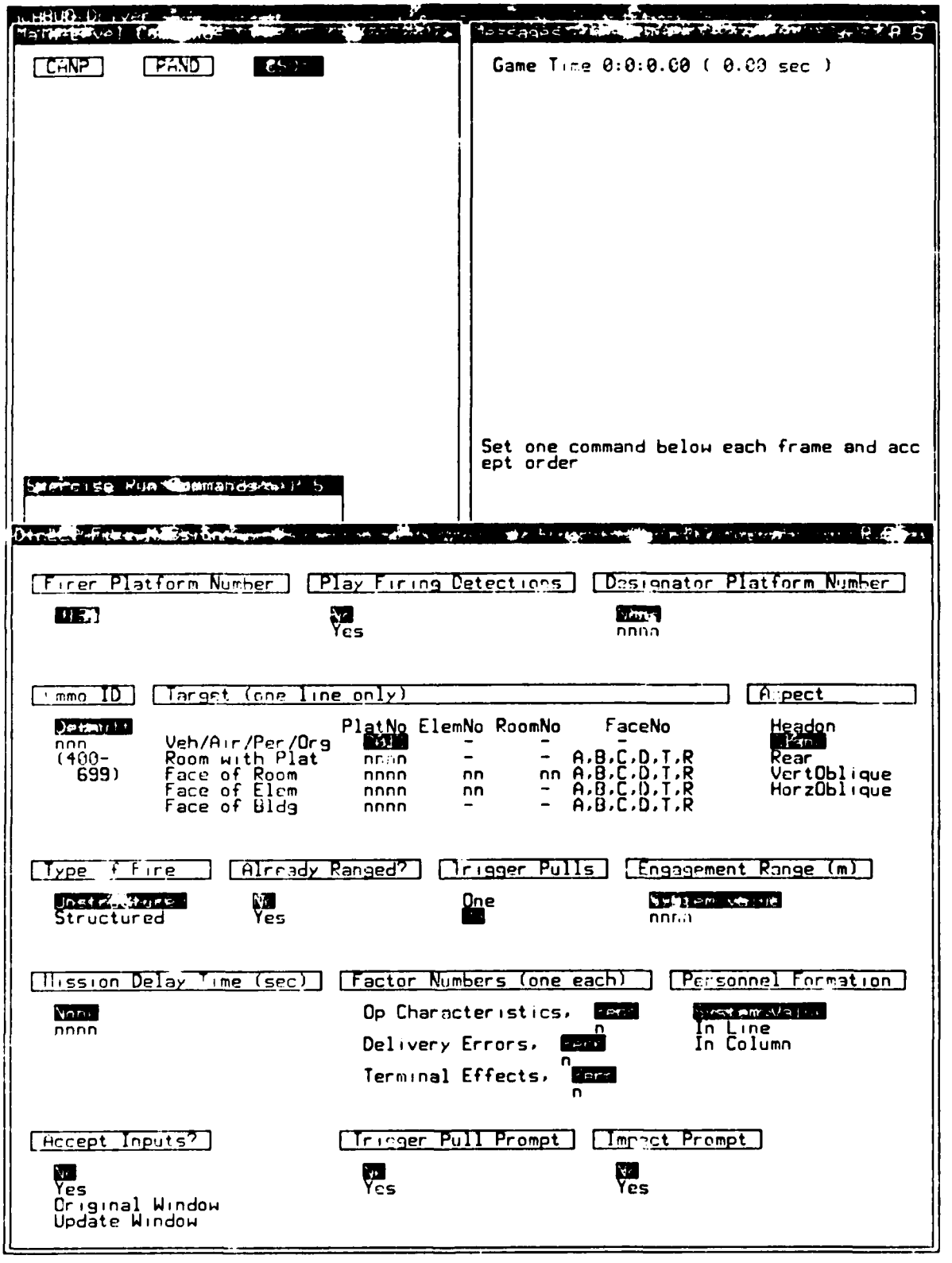


FIGURE 7. Direct Fire Order Window

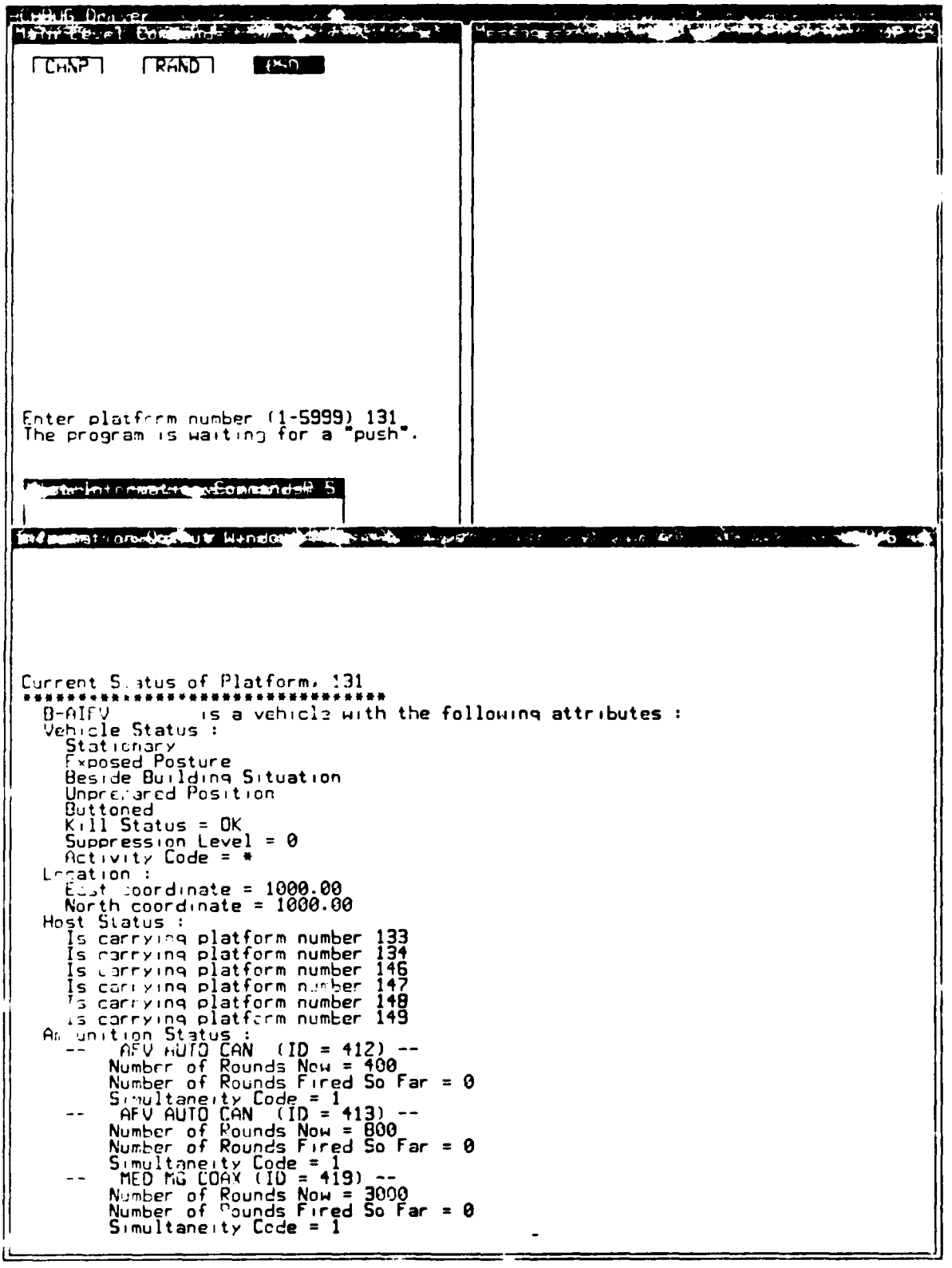


FIGURE 8. List Information and Example Information Windows

(b) Original Window. An original window is defined in the code for each order, i.e., the standard defaults are set and all readable commands are delimited using strings of "n". When orders are input, these original commands are changed. To aid in the input of similar orders, the last status of the commands in that window is displayed when the window is reactivated for the next order of the same type. Sometimes, however, it is easier to start from the original commands. Thus, a "push" on the original window command will restore the original command state.

(c) Update Window. This command is used to erase any error messages in the window and to restore any commands they may have overwritten. It restores all commands in the window to their current state in the model.

e. Optical Mark Document Reader.

(1) It would be undesirable for the "command" package to be the only form of input as it could create a bottle-neck at the PERQ terminal. Hence players can enter the more common orders by filling out special optical mark reader (OMR) input forms which are then read by a Scan-Tron optical mark document reader.

(2) For cost reasons, only six OMR input forms are currently available:

- o Direct Fire
- o Ground Movement
- o Indirect Fire
- o Mount/Dismount
- o Non-firing Target Acquisition
- o Status Update

However, in principle, any number of such forms could be prepared. Examples for Indirect Fire and Ground Movement are shown in Figures 9 and 10.

(3) The forms are completed by marking the appropriate bubbles with a pencil. Most boxes on the form usually have defaults which are used if no entry is made in that box. If an error is detected when reading a form (e.g., a box is only partly completed) then a message is displayed on the PERQ terminal together with an indication of what has, and has not, been successfully read.

5. OUTPUT

a. It will be noted from Figures 5-8 that the PERQ terminal is composed of three major window areas as follows:

GENERATE INDIRECT FIRE MISSION

FIRER PLATFORM

⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃
⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃
⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃
⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃
⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃
⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃
⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃
⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃
⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃
⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃

COMMO NET

⊂0⊃	⊂0⊃	⊂0⊃
⊂1⊃	⊂1⊃	⊂1⊃
⊂2⊃	⊂2⊃	⊂2⊃
⊂3⊃	⊂3⊃	⊂3⊃
⊂4⊃	⊂4⊃	⊂4⊃
⊂5⊃	⊂5⊃	⊂5⊃
⊂6⊃	⊂6⊃	⊂6⊃
⊂7⊃	⊂7⊃	⊂7⊃
⊂8⊃	⊂8⊃	⊂8⊃
⊂9⊃	⊂9⊃	⊂9⊃

AMMO I.D.

⊂0⊃	⊂0⊃	⊂0⊃
⊂1⊃	⊂1⊃	⊂1⊃
⊂2⊃	⊂2⊃	⊂2⊃
⊂3⊃	⊂3⊃	⊂3⊃
⊂4⊃	⊂4⊃	⊂4⊃
⊂5⊃	⊂5⊃	⊂5⊃
⊂6⊃	⊂6⊃	⊂6⊃
⊂7⊃	⊂7⊃	⊂7⊃
⊂8⊃	⊂8⊃	⊂8⊃
⊂9⊃	⊂9⊃	⊂9⊃

MISSION PRIORITY

⊂0⊃	⊂0⊃	(Default is zero)
⊂1⊃	⊂1⊃	
⊂2⊃	⊂2⊃	
⊂3⊃	⊂3⊃	
⊂4⊃	⊂4⊃	
⊂5⊃	⊂5⊃	
⊂6⊃	⊂6⊃	
⊂7⊃	⊂7⊃	
⊂8⊃	⊂8⊃	
⊂9⊃	⊂9⊃	

NUMBER OF VOLLEYS

⊂0⊃	⊂0⊃	(Default is one)
⊂1⊃	⊂1⊃	
⊂2⊃	⊂2⊃	
⊂3⊃	⊂3⊃	
⊂4⊃	⊂4⊃	
⊂5⊃	⊂5⊃	
⊂6⊃	⊂6⊃	
⊂7⊃	⊂7⊃	
⊂8⊃	⊂8⊃	
⊂9⊃	⊂9⊃	

TARGET TYPE *Choose one only of Board Coordinates, UTM Coordinates, Known Pt or Platform*

Board Coordinates (m) ⊂ ⊃
UTM Coordinates m ⊂ ⊃

Easting (⊂—⊃)						Northing (⊂—⊃)							Known Pt Number				Platform Number			
⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃
⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃
⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃
⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃
⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃
⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃
⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃
⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃
⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃
⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃

OR

COMMO CONDITIONS

[illegible]	(Default)
[illegible] EW	⊂ ⊃
High EW	⊂ ⊃
Comm Blackout	⊂ ⊃

FUZE TYPE

Point Detonating	(Default)
AirBurst	⊂ ⊃
Delay	⊂ ⊃

ANGLE OF FIRE

Low	(Default)
High	⊂ ⊃

ACCURACY OF FIRE

K Transfer	(Default)
Unadjusted	⊂ ⊃
Adjusted	⊂ ⊃

DESIRED IMPACT GAME TIME

ASAP or (Default)

HOURS : MINUTES

⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃
⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃
⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃
⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃
⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃
⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃
⊂6⊃	⊂6⊃	⊂6⊃	
⊂7⊃	⊂7⊃	⊂7⊃	
⊂8⊃	⊂8⊃	⊂8⊃	
⊂9⊃	⊂9⊃	⊂9⊃	

SHEAF TYPE

Normal with Firing Pattern Number (Default is zero)

	⊂1⊃
	⊂2⊃
or	⊂3⊃
Converge	⊂4⊃
⊂	⊂5⊃
	⊂6⊃
	⊂7⊃
	⊂8⊃
	⊂9⊃

FORWARD OBSERVER NUMBER

None or (Default)

⊂0⊃	⊂0⊃	⊂0⊃	⊂0⊃
⊂1⊃	⊂1⊃	⊂1⊃	⊂1⊃
⊂2⊃	⊂2⊃	⊂2⊃	⊂2⊃
⊂3⊃	⊂3⊃	⊂3⊃	⊂3⊃
⊂4⊃	⊂4⊃	⊂4⊃	⊂4⊃
⊂5⊃	⊂5⊃	⊂5⊃	⊂5⊃
⊂6⊃	⊂6⊃	⊂6⊃	⊂6⊃
⊂7⊃	⊂7⊃	⊂7⊃	⊂7⊃
⊂8⊃	⊂8⊃	⊂8⊃	⊂8⊃
⊂9⊃	⊂9⊃	⊂9⊃	⊂9⊃

OPERATIONAL CHARACTERISTICS FACTOR NUMBER

(Default)
⊂1⊃
⊂2⊃
⊂3⊃
⊂4⊃
⊂5⊃
⊂6⊃
⊂7⊃
⊂8⊃
⊂9⊃

DELIVERY ERRORS FACTOR NUMBER

(Default)
⊂1⊃
⊂2⊃
⊂3⊃
⊂4⊃
⊂5⊃
⊂6⊃
⊂7⊃
⊂8⊃
⊂9⊃

TERMINAL EFFECTS FACTOR NUMBER

(Default)
⊂1⊃
⊂2⊃
⊂3⊃
⊂4⊃
⊂5⊃
⊂6⊃
⊂7⊃
⊂8⊃
⊂9⊃

INDIRECT FIRE

INSTRUCTIONS

1. Make response marks complete.
2. Make full erasures of revised responses.
3. Erase stray or unintentional marks.
4. All numbers must be right-justified.
5. There should be an entry in each box unless there is a default indicated. Putting an explicit entry in a box overrides the default. Shaded areas (Firer Platform, Commo Net, Ammo I.D. and Target Type) do not have defaults and must be entered.
6. When entering a number, a mark must be placed in every column, i.e., leading zeros must be explicitly indicated.
7. A negative board coordinate is indicated by filling in the bubble marked (⊂—⊃).

USE NO. 2 PENCIL ONLY

FEED THIS DIRECTION

FIGURE 9. Optical Mark Reader Form for Indirect Fire

VEHICLE / PERSONNEL / ORGANIZATION MOVEMENT

PLATFORM NUMBER

0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9

FORMATION TYPE

None (Default)		
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9

MOTION TYPE

Normal (Default)
Erase

PROMPT IF MOUNTED?

Yes (Default)
No

MOVEMENT TYPE (Choose One)

Time Lapse (sec)	Constant Vel (km/hr)	Arrive Time Hours	Minutes
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	
7	7	7	
8	8	8	
9	9	9	

PERSONNEL

DURING MOVE		
Posture	**Situation**	**Protection**
Standing (Default)	General (Default)	Unprep (Default)
Crouching	Beside Bldg	Prep No Over
Prone	Behind Bldg	Prep With Over

END OF MOVE		
Posture	**Situation**	**Protection**
Standing (Default)	General (Default)	Unprep (Default)
Crouching	Beside Bldg	Prep No Over
Prone	Behind Bldg	Prep With Over

VEHICLES

DURING MOVE		
Posture	**Situation**	**Protection**
Exposed (Default)	General (Default)	Unprep (Default)
Hull Def	Beside Bldg	Prep No Over
Vert Def	Behind Bldg	Prep With Over
Turret Def		

END OF MOVE		
Posture	**Situation**	**Protection**
Exposed (Default)	General (Default)	Unprep (Default)
Hull Def	Beside Bldg	Prep No Over
Vert Def	Behind Bldg	Prep With Over
Turret Def		

ACTIVITY CODE

AAAA (Default)

1st Letter	A,B,C	D,E,F	G,H,I	J,K,L	M,N,O	P,Q,R	S,T,U	V,W,X	Y,Z	1	2	3
2nd Letter	A,B,C	D,E,F	G,H,I	J,K,L	M,N,O	P,Q,R	S,T,U	V,W,X	Y,Z	1	2	3
3rd Letter	A,B,C	D,E,F	G,H,I	J,K,L	M,N,O	P,Q,R	S,T,U	V,W,X	Y,Z	1	2	3
4th Letter	A,B,C	D,E,F	G,H,I	J,K,L	M,N,O	P,Q,R	S,T,U	V,W,X	Y,Z	1	2	3

DESTINATION (Choose One)

Board Coordinates (m)

Easting (-)					Northing (-)				
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

Known Point

0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9

Platform

0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9

UTM Coordinates (m)

Easting						Northing						
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9

MISSION DELAY TIME (sec)

None (Default)			
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9

USE NO 2 PENCIL ONLY

SCAN-TRON

FORM NO. 3104-USA

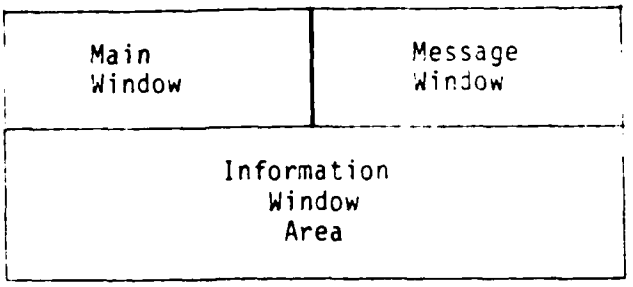
830524

DIRECTION FEED THIS

INSTRUCTIONS

1. Make response marks complete.
2. Make full erasures of revised responses.
3. Erase stray or unintentional marks.
4. All numbers must be right-justified.
5. When entering a number, a mark must be placed in every column, i.e., leading zeros must be explicitly indicated
6. There should be an entry in each box unless there is a default indicated. Putting an explicit entry in a box overrides the default.
7. Shaded areas do not have defaults and must be entered; however, only one of the four destinations and only one of the three movement types can be entered.
8. A negative board coordinate is indicated by filling in the bubble marked (-).
9. Formation type applies only to organization movement. The posture, situation and protection for both personnel and vehicles may be indicated (if appropriate) for organization moves.
10. Activity code letters are indicated by marking the appropriate letter group and 1, 2 or 3 to show the letter's position within the group.

FIGURE 10. Optical Mark Reader Form for Ground Movement



b. The main window is used to display:

(1) Routine operator information to assist the mechanics of running an exercise.

(2) Information which may be beneficial for the controller to know, but which he may not wish the players to know. Typically this is information which is being displayed ahead of time.

c. The message window is used to display regular information that is intended for the players. For an open game this portion of the screen can be televised using a video camera feeding to remote monitor displays. The frequency with which the PERQ screen is refreshed is sufficiently high to enable a flicker-free display to be produced.

d. The information window area has a dual purpose.

(1) Windows for the various input orders use this area.

(2) Non-routine output is presented here, e.g., displays of the future event list, or the status of a particular platform.

e. The PERQ can be connected to a Canon laser printer, and the facility to use such a printer is included in the software. The two options available are:

(1) List any text file residing on the hard disk on the printer. Hard copy is then produced at 60 lines/page and 10 pages/minute. This action is initiated with a "press" on the CANP command in the main window.

(2) Duplicate the screen at any stage of the exercise except when the computer is in the middle of an assessment. To duplicate the screen all that is required is for the "puck" to be pressed when the cursor is positioned on the CSDX box in the main window. A hard copy of the current display is available in approximately 24 seconds.

f. For use with the manual rules, a linear congruential multiplicative random number generator (as defined in Reference 9) is provided. A "press"

on the RAND command in the main window gives a uniform random number between zero and one in the message window.

6. THE FUTURE EVENT HANDLER

a. Most assessments in ACABUG use discrete event simulation techniques, so that a time-ordered list of future scheduled events must be maintained. The ACABUG future event handler is comprised of two major parts:

(1) A linked list (using PASCAL pointer variables) of future events held in main memory. This list is time ordered, so that the head of the list is always the next event to be processed.

(2) A doubly linked file of future events held in a disk file. This file is not time ordered, the links being used only to handle file storage.

The linked list of future events is adapted from Reference 6 and the future event file is adapted from Reference 7.

b. The Linked List Of Future Events.

(1) A linked list of future scheduled events is kept in main memory for speed of access. This is a singly linked list, with earlier events in time preceding later events. Use is made of PASCAL pointer variables to create and maintain this list. To conserve memory, the list does not contain the full set of information needed to process an event during an exercise run. Rather, each entry contains just a skeleton set of information, together with a pointer to the full set of information which is stored in the event file (see below).

(2) The skeleton information stored in the linked list is as follows:

(a) EventTime. The time for which the event is scheduled.

(b) IndexNr. The record (slot) number in the event file at which the full set of information for this event can be found.

(c) EventData. A PASCAL record containing:

- The type of event
- The source platform number
- The target platform number
- An ancillary integer flag
- The mission number. A unique number for each mission is allocated so that events can easily be related to specific input orders.

c. The Event File. The event file is a random access file, with a double set of integer pointers to assist file management. This file serves two purposes.

(1) It saves having to store the full set of event information in the linked list of future events, thus saving main memory storage.

(2) It provides a safety backup in case of a system failure or power down. The event file contains all the information needed to recover the event list, even if that list has been completely lost (each time a "hot start" is made during the course of an exercise run, the linked list is re-created from the event file).

d. The Event Handler Procedures. There are seven major procedures available to the user.

(1) InitEventListAndFile. This procedure initializes the event list and file. It creates an empty event file with a "zero" information record, and an empty event list with a "nil" pointer. This procedure should only be used at the start of a completely new exercise run as it erases any existing event file or list.

(2) RecoverEventList. This procedure recovers the linked list of future scheduled events (the event list) corresponding to an existing event file. Any existing event list is destroyed, but the event file is unaffected. This procedure should be called at the start of each day of an exercise run, except the first when InitEventListAndFile should be called. It should also be called when resuming an exercise run after a system error to ensure that the event file and list are properly synchronized.

(3) AddNewEvent. This procedure adds a new event to the event file and inserts it also in the event list as specified by the time for which the event is scheduled.

(4) GetNextEvent. This procedure gets the next event in time from the event list and returns the full set of information stored for that event in both the event list and file. The event is also deleted from both the event list and the event file. If more than one event is scheduled for the same time they are removed from the head of the event list in the same order as they were added to the list (i.e., the list acts like a first-in first-out queue).

(5) ListFutureEvents. This procedure writes a list of some or all of the events in the future event list to a text file, which may then be either printed or displayed on the PERQ screen. There are logic options available to control what events are listed.

(6) DisplayEventList. This procedure displays on the PERQ screen a list of future scheduled events that has been previously generated by the ListFutureEvents procedure.

(7) DeleteEvents. This procedure deletes events from both the event list and event file. Similar options to control which events are to be deleted are provided as for the ListFutureEvents procedure.

7. THE SCHEDULING AND ASSESSMENT ALGORITHMS

a. During an ACABUG exercise the user interacts with the computer software primarily through the "run-time menu" which contains the following options:

Air Movement
Battle Log Entry
Building Clearance
Communications
Direct Fire
Ground Movement
Indirect Fire
LOS
Minefields (not yet implemented)
Mount/Dismount
Non-firing Target Acquisition
Optical Mark Reader Input
Event List Modification
Game Parameter Update
List Information
Status Update
Execute Next Event
Execute to Game Time
Execute n Events

b. All of these options, except for the "execute" options, leave game time unchanged and simply add further orders to the event list, interrogate and update data files or provide displays on the screen. Game time is only ever advanced when the operator selects one of the "execute" options.

(1) Execute Next Event. Execute only the next event in time from the head of the future event list. Game time is advanced to the time of that event. Then, depending on the type of event, any necessary assessments are made before control is returned to the run-time menu. This is the normal mode of time advance.

(2) Execute to Game Time. The operator is prompted to enter a game time. The model will then execute the events in the future event list in turn until the next event's time exceeds the inputted game time. Game time is advanced on an event basis accordingly, stopping on the time of the last event actually executed. It may be necessary to answer prompts during these executions, depending on the type of events being executed. However, the operator cannot regain control until the time period is completed.

(3) Execute n Events. The operator is prompted to enter an integer number of events to be executed from the future event list. Again each event

is executed in turn (each possibly requiring answers to prompts) and game time stops at the time of the last event actually executed. The operator cannot gain control until all "n" events are executed.

c. After the last event is executed, control is returned to the run-time menu. The operator may then elect to input further orders, delete some existing orders, alter the status of a unit, display some information, or execute one or more events, in which case game time is further advanced. Typically one event at a time will be executed to allow the above mentioned actions to take place between advances in game time. Once game time has advanced, it is not possible to back it up. Play can be restarted, however, from any point where the dynamic game files have been saved. It takes approximately ten minutes to copy the dynamic game files to floppy disk.

d. The two basic types of units in ACABUG for assessment purposes are platforms and ammunition.

(1) Platforms.

(a) Platforms can be generally thought of as physical entities such as tanks or personnel, and can usually:

- o Carry other platforms
- o Carry and fire ammunition
- o Be killed or damaged.

(b) In ACABUG the following types of platforms are explicitly represented:

- o Vehicles
- o Personnel
- o Airborne
- o Buildings
- o Element (sub-section of a building)
- o Room
- o Sensor
- o Organizational
- o Man-pack

At the time of writing man-pack platforms are not properly supported and are really included for possible future expansion.

(2) Ammunition

(a) Ammunition may be carried and fired by platforms. A single platform may carry several different types of ammunition, and ammunition may have separate direct and indirect fire characteristics, if appropriate.

(b) Ammunition per se cannot be killed or damaged, but if the platform that is carrying it has an appropriate type of kill (e.g., F-kill) then it will be prevented from firing the ammunition.

e. In order to increase the speed of play and ease the burden placed on players, organizational platforms have been introduced. These are conceptual rather than physical platforms; they have subordinates which may be physical platforms and/or other organizational platforms. The main purpose of introducing these platforms is that a single order issued to such a platform will automatically be obeyed by all its subordinates. Thus a player can order for example, a whole squad to fire with just a single order. Even though the order is issued to the squad, the assessment still takes place at the individual platform level, thus maintaining the high resolution of the game.

8. NON-FIRING TARGET ACQUISITION

a. Description

(1) The non-firing target acquisition command determines whether a detection attempt is successful, and if so the time at which the target is detected.

(2) Multiple detection attempts can be handled using organizational platforms. In this case the input procedure breaks the "compound order" down into a series of 1-on-1 independent detection attempts, which are then "bracketed together" to show that they all belong to the same compound order.

b. Inputs. A non-firing target acquisition order has the following inputs:

(1) Delay time (secs) between input of the order to the computer and commencement of the detection assessment.

(2) Platform number of observer (may be an organizational platform).

(3) Sensor ID code (if different from the one associated with the platform ID code). The default option is to use the observer platform ID code to determine detection probabilities.

(4) Platform number of target (may be an organizational platform).

(5) Factor number. This is used to enable different sets of conditions to be handled other than those explicitly catered for. The default value is zero.

(6) Target aspect (Head-on, Flank, Rear, Vertical Oblique). The default value is Head-on. This entry is ignored if the target is a personnel platform. The Vertical Oblique aspect is for vehicles only.

(7) Search type (Surveillance, Local Search, Stare).

(8) Background type (Fields, Trees, Urban, Inside Building, Sky/Smoke).

(9) Obscuration (Clear, Light, Medium, Heavy). When a dynamic obscuration model is implemented the obscuration level will then be calculated automatically and this entry will no longer be a player input.

(10) Look ahead time (secs). Only detections with delays less than the look ahead time are to be considered successful. The default option is to consider only detections up until the next terrain board update.

9. FLASH DETECTION

a. Description.

(1) The flash detection command determines whether a firing signature detection attempt is successful, and if so, whether the target is accurately or only approximately located. In addition a non-firing detection attempt may also be automatically initiated, depending on the value of a switch contained in the exercise setup conditions.

(2) The command can handle detecting platforms that are organizational platforms, in which case the input procedure breaks the command down into a series of 1-on-1 detection attempts.

(3) The command is not called directly from the run-time menu. Instead the "play flash detectors" option must be specified on the relevant direct fire order, in which case the flash detection command will automatically be initiated when the weapon fires. Note that if the "play flash detectors" option is specified on a direct fire order to an organizational platform, the direct fire input procedure breaks that order down into a series of 1-on-1 fire orders, each with the "play flash detectors" option on. Thus flash detection may involve organizational observers, but not organizational firers.

b. Inputs

(1) The flash detection input procedure is (optionally) automatically initiated when a direct fire trigger pull event is executed. The following inputs are obtained from information stored for that event, and hence do not need to be entered via the terminal:

(a) Firer platform number.

(b) Ammo ID code.

(2) The following inputs must be entered via the terminal by the operator:

(a) Platform number of observer (may be an organizational platform).

(b) Sensor ID code (if different from the one associated with the platform ID code). The default option is to use the observer platform ID code to determine detection probabilities.

(c) Factor number. This is used to enable different sets of conditions to be handled other than those explicitly catered for. The default value is zero.

(d) Background type (Fields, Trees, Urban, Inside Building, Sky/Smoke).

(e) Obscuration (Clear, Light, Medium Heavy). When a dynamic obscuration model is implemented the obscuration level will be calculated automatically and this entry will no longer be a player input.

(f) Whether the observer is previously aware of the presence of the target (e.g., has previously obtained a localized detection). The answer is Yes or No, the default being No.

(3) The input procedure starts by asking if any more observers are to be considered. A "no" response exits the flash detection routine. A "yes" response causes a prompt for the operator to make the necessary inputs, after which the assessment(s) are made. The input procedure then asks again if any more observers are to be considered, and the loop is repeated. It should be noted that if an organizational platform is input as the observer, then the assessment algorithm will automatically loop over all subordinate platforms before returning to ask if any more observers are to be considered.

(4) If the flash detection assessment causes a non-firing detection attempt to be triggered then the factor number, sensor ID code, background and obscuration are the same in both cases. In addition the non-firing detection attempt uses:

(a) A default aspect of Head-on (for vehicle and airborne targets only).

(b) The look head time as specified in the exercise setup conditions.

(c) A search mode of:

o Stare following an accurate flash detection.

- o Local Search following a localized flash detection.

- o Surveillance following a failed flash detection.

10. COMMUNICATIONS

a. Description. The communications command determines whether a "message" is successfully passed or transmitted over a specified "virtual net," and if successful then the time at which the "message" is received and acted upon. Note that this is a generalized concept of message and communication net. The transmission of a "message" over a "virtual net" may include much more than the physical transmission of the message by wire, radio, etc. It may include the complete communications process, covering the duration from the moment the "message" is first initiated until the moment it is acted upon, including queueing times, transmission times, decision making times, etc.

b. Inputs. A communications order has the following inputs:

(1) Delay time (secs) between input of the order to the computer and commencement of the communications assessment.

(2) "Virtual net" number.

(3) Communications conditions (No EW, Low EW, Moderate EW, High EW).

(4) "Message" to be transmitted.

11. INDIRECT FIRE

a. Description

(1) The indirect fire command schedules and executes indirect fire missions. A single input command can order a unit to fire more than one volley, but all the volleys must have the same aimpoint.

(2) Indirect fire orders can be given to organizational platforms, in which case they are relayed to and executed by each subordinate platform, (which may in turn be an organizational platform). Thus if an order to fire m volleys is given to an organizational platform with n physical subordinate platforms, this causes each of the subordinates to fire m volleys.

(3) Only a single aimpoint may be specified for a single fire order. However, when the order is given to an organizational platform, and patterned fire is specified, this will automatically cause the individual gun aimpoints to be appropriately displaced about the organizational (e.g., battery) aimpoint without further user intervention.

(4) There are two alternative methods for processing indirect fire missions. The choice is governed by the assessment code of the firing platform, and depends on how indirect fire resources are to be represented in the model.

(a) "Real Weapon" Option. With this option, each weapon as stored within the computer, is assumed to correspond with a "real" weapon on the battlefield. Thus the weapon can only fire one mission at a time, and the software automatically enforces this rule. If more missions than the firing platform can handle are input to the computer, the outstanding missions are queued, to be fired when the platform becomes available. This option is appropriate for mortars deployed in direct support of the forces portrayed in the exercise scenario, and perhaps also direct support artillery.

(b) "Virtual Weapon" Option. With this option each weapon, as stored within the computer, does not necessarily correspond with any specific "real" weapon actually deployed on the battlefield. Instead, they should be considered as representative weapons from a larger pool of resources. Thus, no restriction is placed on how many missions a "virtual weapon" can fire at any one time. It is realized that it may be impossible for a single weapon or battery to actually fire several interleaved missions in parallel; but it should be interpreted as several missions coming possibly from several different sources. This option may be appropriate for general support artillery. The possible non-availability or delayed availability of the indirect fire support is modeled in ACABUG by transmitting the order over a "virtual" commo net with appropriate characteristics. It is, of course, possible to represent both "real" and "virtual" weapons using only this second option, but in that case it becomes a manual controller responsibility to prevent unrealistic multiple missions being executed by "real" weapons.

b. Inputs. The following inputs are required for indirect fire orders:

(1) If a specified impact time is wanted, then the game time (in secs) at which it is desired for the first round to impact. Otherwise it is assumed that the missions should be executed as soon as possible.

(2) "Virtual" net number (900 - 949) over which the order must be passed to the firing unit.

(3) "Commo" conditions (No EW, Low EW, Moderate EW, Heavy EW).

(4) Firing platform number.

(5) Forward observer number (if appropriate).

(6) Ammo ID code.

(7) Fuze option (Point Detonating, Airburst, Delay).

(8) High or Low angle fire.

(9) Factor number for operational characteristics. The default value is zero. This factor number enables factors which affect the operational characteristics of the firing platform, but which are not explicitly modeled, to be handled.

(10) Factor number for delivery errors. The default value is zero. This factor number enables factors which affect delivery errors, but which are not explicitly modeled, to be handled.

(11) Factor number of terminal effects. The default value is zero. This factor number enables factors which affect terminal effects, but which are not explicitly modeled, to be handled.

(12) Number of volleys (trigger pulls) to be fired, by each subordinate platform in the case of orders to organizational platforms.

(13) Target type.

(a) Local x-y coordinates, consisting of an easting coordinate and a northing coordinate, each given in terms of local board coordinates relative to the southwest corner of the terrain boards.

(b) UTM x-y coordinates, consisting of an easting coordinate and a northing coordinate which are converted internally by the computer to the corresponding local x-y coordinates.

(c) Known point number, which is converted internally by the computer to the corresponding local x-y coordinates.

(d) Platform number, which is converted internally by the computer to the corresponding local x-y coordinates for that platform location.

(14) Sheaf type (patterned or converged fire). This only has any effect for orders to organizational platforms with several subordinate firing platforms.

(15) Firing pattern number. The default value is zero. This only has any effect when the type of fire specified in paragraph (14) above is patterned.

(16) Accuracy of fire (Unadjusted,Adjusted,KTransfer). The default is KTransfer.

(17) Mission priority. The default value is zero. High numbers have priority over low numbers. This only has any effect for fire orders to "real" weapons, since "virtual" weapons have an unlimited firing capacity.

12. DIRECT FIRE

a. Description

(1) The direct fire command schedules and executes direct fire missions. A single input command can order a unit to fire more than one round (trigger pull).

(2) Direct fire orders can be given to organizational platforms, in which case they are relayed to and executed by each subordinate platform (which may in turn be an organizational platform). Thus if an order to fire m rounds is given to an organizational platform with n physical subordinate platforms, this causes each of the subordinates to fire m rounds.

(3) Direct fire orders can specify a variety of different target types as follows:

(a) Vehicle, personnel or airborne platform. In this case only the specified platform is assessed as a target, i.e., a "miss" cannot affect other platforms.

(b) Room in a face of a building. In this case if the round misses the specified room it may still hit another room in the same face of the same building (but not any other face of this building, nor any other building). The assessment is made against the room hit (if any).

(c) Room occupied by a specified platform. This is identical to option (b) except that instead of specifying the room number directly, it is specified indirectly via the platform number of its occupying platform. Note that a building face must still be specified.

(d) Face of element in building. In this case a random room from the specified face of the element is chosen as the target room, and the assessment is then the same as for option (b). A new room is chosen for each separate trigger pull.

(e) Face of building. In this case a random element is chosen from the specified face of the building. The assessment is then the same as for option (d). A new element is chosen for each separate trigger pull.

(f) Organizational platform. In this case the assessment depends on whether the platform is declared to be an engageable unit.

- o Engageable Units. These are interpreted as small infantry units, such as a squad (section), fire team or similar, and an assessment is made against the platform as a whole, i.e., against all its subordinate platforms. It should be noted that the assessment procedure does not treat the subordinate platforms independently of each other, and is appropriate only for infantry units.

- o Non-Engageable Units. If the platform is not engageable as a single unit then a random subordinate platform is chosen from the list of immediately subordinate platforms, and the assessment is made against that platform, i.e., an option (a) assessment is made if the subordinate is a vehicle, personnel or airborne platform, or another option (f) assessment is made if the subordinate is itself an organizational platform.

(4) In the case when the target is a building, element or room (options (b)-(e) in the preceding paragraph), the player may elect to specify which face of the building is to be engaged. Alternatively, the player may use the random face option; in this case, a face is chosen at random from those building faces that can be feasibly engaged given the specific geometry of the engagement.

(5) The decision is also made on the direct fire order whether to play flash detection. A positive decision will result in the flash detection routine being initiated each time a platform fires as a result of this fire order. For further details of the flash detection procedures see Paragraph 9.

b. Inputs. The following inputs are required for direct fire orders:

(1) Delay (secs) between input of the fire order to the computer and the start of the direct fire scheduling algorithm. The default option is zero delay.

(2) Firing platform number.

(3) Designator platform number (if needed).

(4) Ammo ID code, unless the default option is selected to allow this to be chosen automatically by the software from the firing platform number.

(5) Target platform number (may be a vehicle, personnel, airborne, building or organizational platform).

(6) Element number (if needed). The element number is relative to the building.

(7) Room number (if needed). The room number is relative to the element.

(8) Face of element or building (if needed). The A, B, D, C, or Top face may be specified, or the random face option may be selected.

(9) Target aspect (Head-on, Flank, Rear, Vertical Oblique, Horizontal Oblique). Horizontal oblique is for rooms, elements and buildings only, not for vehicles or airborne platforms. Target aspect is not used if the target is a personnel platform.

(10) Number of trigger pulls to be fired (by each subordinate platform in the case of orders to organizational platforms). The default is one.

(11) Structured or Unstructured shot.

(12) Engagement range (meters). If none is specified then the engagement range is calculated automatically based on the firer and target locations in the computer. It should only be necessary to specify an engagement range

for very short range weapons such as hand grenades, when the system value may be too inaccurate because of the lack of precision with which unit locations are transferred from the terrain boards to the computer.

(13) Factor number for operational characteristics. The default value is zero. This factor number enables factors which affect the operatonal characteristics of the firing platform, but which are not explicitly modeled, to be handled.

(14) Factor number for delivery errors. The default value is zero. This factor number enables factors which affect delivery errors, but which are not explicitly modeled, to be handled.

(15) Factor number for terminal effects. The default value is zero. This factor number enables factors which affect terminal effects, but which are not explicitly modeled, to be handled.

(16) Play flash detection (yes/no).

(17) Ranging already done (yes/no).

(18) Verify via the PERQ terminal before executing DFTriggerPull events, i.e., before firing each round (yes/no).

(19) Verify via the PERQ terminal before executing DFShotReachesTarget events, i.e., before making a casualty assessment (yes/no).

13. SUPPRESSION

a. ACABUG does not use a "standard" suppression model. Rather it uses a suppression model which is based on ideas taken from a number of other models (particularly the Litton model, Ref 8), and their logical extensions. The structure of the model was dictated largely by the inputs available within the ACABUG software, which are typically kill probabilities (broken down into probability of hit and the conditional probability of kill given a hit for direct fire engagements). In particular, miss distances are not routinely available for all engagements.

b. The suppression model that has been developed is plausible but has not been validated. The issue of validity, and the precise definition and consequences of suppression, are mediated by giving the responsibility of decision to the players. Thus the purpose of the ACABUG suppression model is to give guidance to the players and controllers so that they can reasonably try to take suppression into account when making fire, movement and detection decisions. The software does not itself initiate any actions as a result of suppression, other than prompting the player when orders are issued to suppressed platforms.

c. In outline the suppression algorithm is as follows:

(1) When a direct or indirect fire assessment is made against a platform, an assessment is made to determine if that platform is suppressed.

(2) If the platform is suppressed then:

(a) The platform's suppression status is incremented by one (an unsuppressed platform's suppression status is zero).

(b) A duration of suppression is sampled, which may depend on the current suppression status of the platform.

(c) An "End Suppression" event is entered into the future event list for the current game time plus the duration of suppression.

(d) A warning message is printed out giving the platform's new suppression status.

(3) When an "EndSuppression" event is executed the platform's suppression status is decremented by one, and a message giving the revised suppression status is output.

d. The model itself takes no actions, nor inhibits any actions, as a result of suppression, although it will print out warning messages if orders are given to suppressed platforms in order to ensure that their suppression status has not been overlooked. However, if the player wishes to proceed with the order, then there is nothing in the software to prevent him so doing.

e. Thus, the onus is placed on the players to make realistic decisions and responses to situations in the light of their military experience. The integer-valued suppression status of a platform is intended to act as a helpful guide for the players; although it is not precisely defined, the general implication is that the higher the suppression status, the greater is the threat being faced by that platform and the greater is likely to be its degree of suppression.

f. The details of the suppression algorithm are given at Appendix A.

14. MOUNTING AND DISMOUNTING PLATFORMS

a. Description

(1) The mount (dismount) command schedules and executes the mounting (dismounting) of platforms on other platforms or in rooms.

(2) Mount/Dismount commands can be given to organizational platforms, in which case they are relayed to and executed by each subordinate platform (which in turn may be an organizational platform). However, it should be noted that after executing a mount command, an organizational platform is not recorded as being itself mounted on the host platform; only its physical subordinates are so recorded.

(3) When platform A dismounts platform B, it will inherit the same location as platform B. In the case of a dismount order to an organizational platform, all the dismounting platforms inherit exactly the same location. If desired, a dispersal radius can be specified, in which case each dismounting platform is randomly located within a circle centered at platform B and radius as specified. This reduces the possibility of all the dismounting platforms being killed by a single "lucky" shot (i.e., a grenade because they happened to have exactly the same coordinates.

b. Inputs

(1) Mount or dismount order?

(2) Mounting or dismounting platform number.

(3) The time lapse or time duration in seconds that it takes to complete the mount or dismount operation.

(4) The new or old host platform as appropriate (i.e., in the case of a mount order the new host on to which the platform is to be mounted). The host may be a vehicle, personnel or airborne platform, or a room.

(5) Orders to mount personnel in rooms only. If the platform is a personnel platform that is being mounted in a room then it can have specified:

(a) A posture (Standing, Crouching, Prone).

(b) A protection (Unprepared, Prepared with No Overhead Cover, Prepared with Overhead Cover).

For any other type of platforms mounting a room, its status must be updated (if needed) by a separate status update order. In the case of personnel mounting a vehicle or airborne platform, their posture and protection becomes irrelevant as their vulnerability is then determined entirely through conditional fractional casualty values.

(6) Dismount orders only.

(a) A dispersal radius may be specified, in which case each dismounting platform is randomly dispersed within a circle of this radius. The default option is no dispersal.

(b) In the case of personnel platforms dismounting from another platform or from a room, they may have specified:

- o A posture (Standing, Crouching, Prone)
- o A situation (General, Beside a Building, Behind a Building)
- o A protection (Unprepared, Prepared with No Overhead Cover, Prepared with Overhead Cover).

(7) The delay time (seconds) between input of the order to the computer and the start of the mount or dismount operation.

15. MOVEMENT (AIR and GROUND)

a. Description

(1) Movement orders may be given to organizational platforms. In this case:

(a) The movement order is executed by the "fictitious" organizational platform itself.

(b) The order is relayed to and executed by all its subordinate platforms (which may themselves be organizational).

This is the only occasion when orders to organizational platforms are also executed by those platforms; normally they are only transmitted to and executed by subordinate platforms. The reason for this is simply to keep the location record of the organizational platform reasonably up to date as a general indication of the group location. The only time organizational platform locations are actually used by the software is during direct fire shots against "engagable" organizational platforms.

(2) In the case of organizational moves a formation number must be specified. This will cause subordinate platforms to be deployed relative to the organizational platform according to the specified formation. Prior to the start of an exercise, a file of desired formations must be established by the user.

(3) At the time of writing, only straight line, constant velocity moves and "instantaneous jump" moves had been implemented. However, expansion of the code to handle multi-segment movement orders would be relatively straightforward, although this would ideally need to be accompanied by a form of graphical input using the PERQ graphics capability for a significant advance in efficiency to be gained.

(4) Data structures and handlers also exist for multi-segment, predefined routes to be established (e.g., following the road network). The intention is to allow the capability of movement orders which simply tell a platform to move along a given route number. At the time of writing, this option was not yet available.

(5) If a movement order is given to a platform that is mounted on another platform then an optional prompt will be displayed, enabling either the order to be aborted or the platform automatically dismounted.

(6) Similar non-optional prompts are also given in the case when:

(a) A movement order is given to a platform which is already executing a movement order. In this case either the new order can be aborted and the existing order continued, or the existing order can be aborted and the new order started.

(b) A suppressed platform tries to start moving. In this case the player has the option of aborting the order or continuing with the move.

(7) If a platform's kill status is such that it would be unable to move, then a warning message is given and the order is automatically aborted.

b. Inputs. The following inputs are required for movement orders:

(1) Moving platform number (may be organizational).

(2) Four character activity code. The meaning of the activity codes must be established in the manual rules. They have no effect whatsoever on the movement order, and are simply stored for reference purposes.

(3) Formation number. This is only used if the moving platform is an organizational platform. It indexes into a user-defined file of formations which specifies how subordinate platforms are to be deployed relative to the organizational platform to which the movement order is given.

(4) Movement Type. The allowable options are

(a) Time Lapse. In this case the platform instantaneously jumps to its new position after the specified time lapse.

(b) Constant Velocity. In this case the platform moves at a constant velocity and in a straight line to the specified destination. The location of the platform is automatically updated at periodic intervals during this move.

(c) Fixed Arrival Time. This option is exactly the same option as (b) above, except that the velocity is automatically calculated by the computer so that the platform reaches its destination at the specified time.

(5) Motion type (Normal or Evasive).

(6) Destination. The destination need not be located on the three-dimensional terrain boards. It may be specified in one of four ways:

(a) Local Board Coordinates. In this case the destination coordinates are given in meters relative to the lower southwest corner of the terrain boards, which is taken as the coordinate origin. It should be noted that within the computer, all locations are stored in this coordinate system.

(b) UTM Coordinates. In this case the destination coordinates are given as full 13 digit (6 digit easting plus 7 digit northing) UTM coordinates.

Note that this is a purely numeric coordinate, and does not use map sheet letters. These coordinates are automatically converted internally to the equivalent local board coordinates.

(c) Known Point Number. This option can only be used provided coordinates for a known point have previously been stored in a data file. The stored coordinates are then substituted for the known point number.

(d) Platform Number. The platform number may be that of a vehicle, personnel, airborne, organizational platform or building. The coordinates of that platform are obtained at the time the order is input. The coordinates so obtained are then taken as the constant destination coordinates. It is important to realize that if the "target" platform subsequently moves, this does not affect this movement order. Once destination coordinates have been obtained they remain fixed throughout that move.

(7) Status During Move. Platforms may be given particular attributes for the duration of the move as follows:

(a) Personnel and Organizational Platforms.

- o Posture (Standing, Crouching, Prone)
- o Situation (General, Beside a Building, Behind a Building)
- o Protection (Unprepared, Prepared Without Overhead Cover, Prepared With Overhead Cover)

(b) Vehicles and Organizational Platforms.

- o Posture (Exposed, Hull Defilade, Vertical Defilade, Turret Defilade).
- o Situation (General, Beside a Building, Behind a Building)
- o Protection (Unprepared, Prepared Without Overhead Cover, Prepared With Overhead Cover)

(c) Airborne and Organizational Platforms.

- o Posture (Exposed, Defilade)
- o Situation (General, Beside a Building, Behind a Building)
- o Altitude (Low, High).

In the case of organizational moves, each subordinate platform extracts from the organizational status the attributes appropriate to its platform type. Note that all subordinates of the same type will thus have identical attributes, so that it is not possible to issue a movement order, for example, to a squad

which will put half the personnel in a standing posture and half in a crouching posture.

(8) Status at End of Move. Platforms may be given attributes to be adopted upon completion of the move. The options are identical with those for during the move (see previous paragraph).

(9) Mission Delay Time. The time delay in seconds between input of the order to the computer and the start of the movement assessment. This enables multi-leg movement orders to be input in one batch as a sequence of straight line moves with appropriate time delays.

(10) Prompt if the Moving Platform is Mounted. If this flag is set to "yes" and if a platform receiving a movement order is mounted on another platform, then a prompt will be issued at the terminal, in case this order has been input by error. The user has the option of aborting the movement order or having the platform automatically dismounted. If the flag is set to "no", then in such a situation, the platform would be automatically dismounted with no prompt.

16. BUILDING CLEARANCE

a. Description

(1) ACABUG is capable of handling a sufficiently large number of platforms that it can represent a reinforced infantry company or an armored battalion in defense against an appropriate threat force. When used at such a level, then it would be undesirable for players to become involved in the detail of building clearance operations. Hence, the building clearance algorithm has been designed as a "black box". The players mount all the personnel (attackers and defenders) in the building and input to the computer the building number and which force is the attacker. Thereafter the computer assesses the progress of the building clearance operation, in a "black box" assessment, printing out at periodic intervals the casualties suffered and the number of rooms cleared. Ammunition usage is also recorded.

(2) It should be noted that the player is not required to move personnel from room to room as the clearance operation progresses. In fact, although he is informed by the computer of the total number of rooms cleared so far, he does not know which specific rooms have been cleared. The computer model implicitily assumes that all personnel within the building take part in the fighting.

(3) The only way that players can influence the course of a building clearance operations is:

(a) To put extra personnel in the building, or take personnel out of the building.

(b) To abort the clearance operation.

(4) It should be noted that even though a building clearance is in progress, the personnel involved may still interact with the remainder of the force in the normal way. In particular:

(a) They may fire out of the building at other targets, although the controller should check that such shots are reasonable, given that the building is being cleared at the same time.

(b) They are subject to lethality and suppression assessments from direct and indirect fire versus the building, again in the normal manner.

(5) At the time of writing no existing building clearance models were known to the authors that could easily be adopted to satisfy the above requirement. Moreover, no body of data exists against which, such a model can be validated, although some such trials are known to be planned for the near future. Hence, it has been necessary to devise from scratch a building clearance model for ACABUG, with very little data on which to base this model.

(6) It should, perhaps, be noted that it would be perfectly feasible to actually play building clearance using ACABUG in conjunction with a detailed, three dimensional model of the building. Movement and direct fire would be assessed using the normal ACABUG facilities for these operations, and the building clearance algorithm would simply not be used. In this way, a body of data could be built up for building clearance operations, which could in time be used to help improve and validate this aggregated building clearance model.

b. Inputs

(1) The only inputs required are:

(a) The number of the building to be cleared

(b) An indication of which is the attacking force (RED or BLUE)

(c) Delay time in seconds between input of the order to the computer and the start of the building clearance assessment.

(2) It should be noted that all occupants of the building (and only those occupants) are included in the building clearance assessment. This implies that:

(a) The approach to the building must be played using the normal ACABUG movement and direct fire orders.

(b) All the relevant personnel should be mounted in rooms in the building before the building clearance order is input to the computer, to ensure they are included in the assessment. It does not matter in which rooms platforms are mounted.

(c) Any direct fire shots at or from the building must be handled as normal direct fire shots.

(d) Extra platforms may be mounted in, or existing platforms may be removed from, the building at any time using the normal mount/dismount command.

17. THE POST PROCESSESOR

a. Description

(1) During the course of an ACABUG exercise a history file is automatically generated for subsequent postprocessing. This is a binary file with system name Sys:Part2>BinFileHistory. Although provision has been made for the postprocessor to be integrated with the remainder of the ACABUG software, and to be called from the main level command window, this has not yet been implemented and the current postprocessor is a stand-alone program.

(2) Use has been made of the module facility allowed in PERQ PASCAL, and also of the record data type inherent to PASCAL, to give the postprocessor considerble flexibility, thus allowing it to be tailored to the aims of any particular study.

(3) Whenever any significant event happens in the ACABUG software, one of a number of procedures is called in the history module, and appropriate information is passed to that procedure. In many cases the information passed is that contained in the future event file for the event in question, although this is not always the case. Typically, the amount of information passed is greater than the amount of information that one would want to routinely store in the history file, so some selection must be made. This is achieved by loading selected information into a case-variant record which is then added to the history file. In many cases the procedure may be just a dummy procedure which takes no action at all and does not write to this history file.

(4) The procedures for writing to and reading from the history file are independent of the exact contents of this case-variant record. Hence, it is an easy matter to change what is stored in the history file to meet the particular aims of any given study. Provided that the information is already contained in parameters passed to the appropriate history procedures the following actions are all that is required:

(a) Change the definition of the record.

(b) Record the procedures in the history module so that they extract the appropriate information. Note that there is no need to change the file handling procedures.

(c) The history file will now store the revised information.

(5) It will of course be necessary to revise the postprocessor to take account of the new information stored in the history file, and these

changes will vary according to the information stored and the statistics it is desired to produce. However, no changes will be needed to the file access procedures.

(6) At the time of writing only an interim postprocessor exists. This is an interactive postprocessor providing a killer-victim scoreboard.

b. Interim Post Processor

(1) The interim postprocessor assumes that all direct and indirect fire events have been recorded in the history file, possibly along with other information. The following main level options are allowed:

(a) Duplicate the screen on the canon laser printer.

(b) Make a history text file from the binary history file. This text file provides a readable history of every direct and indirect fire event in the history file.

(c) List the history text file produced by option (b) above on the PERQ screen.

(d) Print any text file on the canon laser printer. In particular the history text file (but not the binary file) produced by option (b) above can be printed using this option.

(e) Merge history files. During the course of an exercise it may be desirable to occasionally dump the history file to a floppy disk and then start a new history file (in order to save disk space). This option enables the separate files so produced to be re-combined into a single history file as needed by the postprocessor.

(f) Make a killer victim scoreboard. This is described more fully below.

(2) The killer victim scoreboard option calls an interactive program allowing the user considerable flexibility in the grouping of killers and victims. The following inputs are required:

(a) A start time (game time in seconds) from which the analysis is to start. The default option is to start from the beginning of the exercise.

(b) An end time (game time in seconds) up to which the analysis is to extend. The default option is to continue until the end of the exercise.

(c) The method by which victim platforms are to be grouped. The available options are:

o By platform number.

o By platform ID code.

o By platform assessment code.

o By platform type.

(d) The method by which the "killers" are to be grouped. The available options are:

o By platform number.

o By platform ID code.

o By platform assessment code.

o By platform type.

o By ammunition ID code.

o By ammunition assessment code.

(e) The kill methods which are to be included in the analysis. The available options are:

o Direct fire only.

o Indirect fire only.

o Mines only.

o Direct and indirect fire only.

o Direct fire and mines only.

o Indirect fire and mines only.

o Direct, indirect fire and mines.

At the time of writing, minefields have not been computerized, so that, there are at present no computer generated mine kills.

(f) Victim Groups.

o If victim groups are to be chosen by platform type then the victim groups are selected automatically by the computer and assigned the names:

oo All BLUE vehicle platforms

oo All BLUE personnel platforms

oo All BLUE airborne platforms

oo All RED vehicle platforms

oo All RED personnel platforms

oo All RED airborne platforms

o For any other option the user must input:

oo The total number of victim groups desired

oo For each victim group in turn:

ooo A group name

ooo The number of entries in the group

ooo The index number, platform number, ID code or assessment code (as appropriate) for each entry in the group.

(g) Firer Groups. The inputs for firer groups are the same as for the victim groups described in the previous paragraph.

(3) The principal outputs are two killer victim scoreboards. An example of this output is at Figure 11.

(a) The first scoreboard excludes any overkills, so that only "first kills" are counted.

(b) The second scoreboard includes all kills, so that overkills are treated in just the same way as first kills.

(4) It should be noted that for both scoreboards:

(a) Any type of kill is counted.

(b) Only computer generated kills appear in the history file, and hence get included in the killer-victim scoreboard.

18. BATTLE LOG

a. Description.

(1) The battle log option allows a player or controller to write text entries into a computerized logbook. This log is saved until the end of the game. It may be referenced at any time during the game.

(2) Each text entry is limited to 240 characters. The wall clock time and the game time are automatically recorded with each entry, along with a unique entry number.

ACABUG KILLER VICTIM SCOREBOARD

Identification Header:
SAMPLE KILLER/VICTIM SCOREBOARD

VICTIM GROUPS:

Group 1 is All Blue vehicle platforms
Group 2 is All Blue personnel platforms
Group 3 is All Blue airborne platforms
Group 4 is All Red vehicle platforms
Group 5 is All Red personnel platforms
Group 6 is All Red airborne platforms

KILLER GROUPS:

Group 1 is All Blue vehicle platforms
Group 2 is All Blue personnel platforms
Group 3 is All Blue airborne platforms
Group 4 is All Red vehicle platforms
Group 5 is All Red personnel platforms
Group 6 is All Red airborne platforms

KILLER VICTIM SCOREBOARD EXCLUDING OVERKILLS

Total number of kills in table = 13

FIRERS	VICTIMS									
	1	2	3	4	5	6	7	8	9	10
1	0	0	0	2	7	0				
2	0	0	0	0	0	0				
3	0	0	0	0	0	0				
4	1	3	0	0	0	0				
5	0	0	0	0	0	0				
6	0	0	0	0	0	0				

KILLER VICTIM SCOREBOARD INCLUDING OVERKILLS

Total number of kills in table = 17

FIRERS	VICTIMS									
	1	2	3	4	5	6	7	8	9	10
1	0	0	0	3	7	0				
2	0	0	0	0	0	0				
3	0	0	0	0	0	0				
4	3	4	0	0	0	0				
5	0	0	0	0	0	0				
6	0	0	0	0	0	0				

VICTIM GROUP MEMBERS CHOSEN BY PLATFORM TYPE

No index numbers are provided in this case

FIRER GROUP MEMBERS CHOSEN BY PLATFORM TYPE

No index numbers are provided in this case

FIGURE 11. Example ACABUG Killer Victim Scoreboard

b. Inputs.

(1) The battle log module is entered by a "push" on the battle log entry command in the exercise run window. The following inputs are then displayed in the message window:

(a) Insert (I). Allows insertion of a 240 character entry.

(b) List (L). Lists the battle log entries into the battle log file, but does not display them on the screen.

(c) Display (D). Displays the battle log entries on the screen as a function of entry number and game time.

(d) Quit (Q). Exits the battle log module.

(2) When the battle log module is exited, the user is prompted that the program is waiting for a "push". At that time any window or command can be selected.

19. LINE-OF-SIGHT (LOS)

a. Description.

(1) The LOS algorithm uses ground elevations at 50 meter grid resolution. It also takes into account rectangular building silhouettes. If a 50 meter grid square is classified as "wooded", a constant vegetation height for that grid square is used in the LOS computation; otherwise, no other vegetation is considered. Also, the microstructure of the buildings (doors, windows, etc.) is not considered.

(2) Because of its limitations, the LOS algorithm is intended to be used only for long lines-of-sight. Most lines-of-sight will still be determined manually by stretching a string.

(3) The result of the LOS computation is returned to the message window.

b. Inputs.

(1) Observer. The observer's local board coordinates (east, north, height) are either entered explicitly or are computed automatically from user inputted UTM coordinates, known point number or platform number.

(2) Target. The target's local board coordinates are entered in the same manner as the observer's.

20. GAME PARAMETER UPDATE

a. Description.

(1) The game parameters are a set of variables which are saved in a file each time one of them is updated. They are necessary for a "hot start" capability, i.e., recovery after a system crash or game stop.

(2) The game parameters include:

(a) Game time in seconds.

(b) Battle log entry counter.

(c) Last mission number.

(d) Terrain board update time in seconds.

(e) Building clearing update time in seconds.

(f) Constant velocity move update time in seconds.

(g) Last random number seed.

(h) Current debug print option number.

b. Inputs.

(1) At the time of this writing, the game parameters can only be changed by exiting the model and editing the file Sys:Part2>FileGameParameters.

(2) Future capability will display the current parameters in the game parameters update window and allow run-time editing of their values through that window using readable commands.

21. LIST INFORMATION

a. Description.

(1) The list information command allows the output of various data and information to the players and controller during the play of the game. The output goes to the information window. Once the information is displayed, it may be put to hardcopy at any time by a "push" on the CSDX command in the main window, which duplicates the current screen display in 24 seconds. Some of the data is also written to system files (e.g., platform locations and subordinate locations) which can then be printed with the CANP command in the main window at a rate of 10 pages/minute.

(2) The current output options are described briefly below.

(a) Event List. This command uses the ListFutureEvents procedure described in paragraph 6.d.(5) to print various subsets of the future event list or the whole list.

(b) Veh/Per/Air/Org Platform Data. This command lists the current status of a platform including its location, host status, motion, posture, situation, protection, kill status, suppression level, activity code, ammunition status and its immediate subordinates.

(c) Room Platform Data. This command lists the current status of a room including its location, damage status, occupation status, whether it has been strengthened or booby trapped and whether it contains a prepared position.

(d) Text File. This command writes any user indicated text file on the computer into the information window.

(e) History File. This command writes the current history file into the information window. The file name Sys:Part2>FileHistory.

(f) Battle Log File. This command writes the current battle log file into the information window. The file name is Sys:Part2>FileBattleLog.

(g) Platform Locations. This command writes the locations (in local board coordinates) in tabular form of a user inputted interval of platform numbers into the information window. If a platform is mounted, the platform number (and, if appropriate, the element and room numbers) of its host is also listed. This table is also written into the file Sys:Part2>FileLocations.

(h) Subordinate Locations. This command writes the same table as above, but is based on a user inputted organizational platform number. All subordinates at every level are included.

(i) Terrain Square Info. This command lists the current information for a user indicated terrain square (50m x 50m), i.e., the ABCA zonal classification, the environment type, the elevation, the buildings in the square (if any) and the obscuration data.

(j) Terrain Board Info. This command lists the current information for a user indicated terrain board (500m x 500m), i.e., the board label, the highest point, the platforms on the board (if any) and whether it has buildings on it. Terrain board indices off the three-dimensional boards can be accessed, but the static board information cannot be provided for them, only the platforms on the board.

(k) Building Occupation Status. This command lists the number of elements in a building and the number of rooms in each of those elements. The ID code for the building and for each element and room is also included. If the building is occupied, the platform number and kill status of each occupant are given on a room basis.

(3) The code and windowing for list information have been structured so that additions and changes can be made relatively easily.

b. Inputs.

(1) The input approach is basically the same for each command, i.e., a "push" on the appropriate command in the list information window and then answering the individual prompts in the main window. The computer will "beep" when it is ready for the next "push" in the list information window.

(2) To allow a CSDX of the information window each time, it is not automatically deactivated. Therefore, an explicit deactivation command is provided in the list information window. A "push" on this command not only deactivates the information window, but also returns control to the exercise run window.

22. STATUS UPDATE

a. Description.

(1) The status update option allows a player or controller to directly modify the current status of a vehicle, personnel, airborne or organizational platform. Inputs are possible through graphical window commands on the screen or through an optical mark reader form. A player is not allowed to change the kill status, zero the suppression status or delete ammunition; therefore, these options are omitted from the optical mark reader form. Only the controller can sanction these changes and they must be done through screen input.

(2) The following status updates are available:

(a) Add a subordinate.

(b) Delete a subordinate.

(c) Change the kill status.

(d) Zero the suppression status.

(e) Change the formation number.

(f) Change the button status.

(g) Change the activity code.

(h) Change the posture.

(i) Change the situation.

(j) Change the protection.

(k) Change the altitude.

(l) Transfer ammunition from one platform to another.

(m) Delete ammunition.

(3) Run-time status updates for buildings, elements and rooms are not available and are not planned. These data changes are intended to be made through the data base editor.

b. Inputs.

(1) Inputs to status update are different from the other orders in that there are no defaults. Only the platform type and platform number must be entered; all other entries are optional. The inputted platform type is correlated with the platform type computed from the platform number. If they do not match, the status update order is rejected. Many status update commands apply only to a specific platform type. These commands are always checked against the inputted platform type.

(2) Any number of status update changes may be input in the same status update order if they apply to the same platform number. The exception is organizational platforms where, for example, the vehicle posture cannot be changed at the same time as the personnel posture. Two different status update orders are needed to do this.

(3) When a status update order is entered, the changes take place immediately, i.e., they occur at the game time indicated in the message window and before any events not yet executed.

(4) The status update window always is activated in its original condition. This is to preclude inadvertent status updates.

APPENDIX A
SUPPRESSION ALGORITHM

1. In some circumstances the ACABUG suppression model requires a probability of suppression P(S) as a function of the "threat level" f. In these cases the model uses the Litton formula (Ref. 3):

$$P(S) = \frac{\exp(\cdot)}{1 + \exp(\cdot)}$$

where

$$\cdot = 10\exp\left[-0.04\ (1 - f)^2/f\right] - 5.0$$

2. The Litton formula contains a parameter, ρ, which was originally intended as a human factors coefficient, taking the value unity for an "average" soldier. However, in ACABUG it is regarded purely as a shaping parameter with its value(s) being given as part of the input data. In fact, different values for ρ may be specified for the following general classes of platform:

a. Heavily Armored Vehicles.
b. Lightly Armored Vehicles.
c. Soft Skinned Vehicles.
d. Personnel.
e. Airborne Platforms.

3. DIRECT FIRE SUPPRESSION

The details of the suppression algorithm vary somewhat between the five classes of platforms described above. In the case of burst fire weapons the assessment applies to each round of the burst. This proviso applied throughout the suppression model.

a. Heavily Armored Vehicles. If the vehicle is not hit, then it is not suppressed. If the vehicle is hit then the probability of suppression is given by the Litton function P(S), with:

f = probability of any type of kill (M, F, or K) given a hit.

b. Lightly Armored Vehicles. Exactly the same as for heavily armored vehicles, except that lightly armored vehicles use a different shaping parameter, ρ, and hence can, if desired, be more susceptible to suppression than their heavily armored counterparts.

c. Soft Skinned Vehicles. If the vehicle is not hit, then the probability of suppression is given by the Litton function P(S) with:

f = probability of any type of kill (M, F, or K), taking into account also the proabability of hit.

If the vehicle is hit, then the assessment is exactly the same as for armored vehicles but using the shaping parameter for soft skinned vehicles.

d. Airborne Platforms. Exactly the same as for soft skinned vehicles, but using the shaping parameter for airborne platforms, and treating the probability of kill as refering to any of attrition kill, forced landing or mission abort.

e. Personnel. If the platform is hit then it is suppressed. If it is not hit, then the probability of suppression is given by the Litton function P(S) with:

f = probability of incapacitation taking hit probability also into account.

4. INDIRECT FIRE SUPPRESSION.

For indirect fire assessments using lethal areas only, an overall probability of kill is available. In all cases the probability of suppression is given by the Litton function P(S) with:

f = probability of any type of kill,

and using the appropriate shaping parameter, γ.

5. SUPPRESSION OF OCCUPANTS

Only suppression of personnel occupant platforms is considered, whether these are mounted in a room or on another platform. For such personnel platforms, the probability of suppression is given by the Litton function P(S) with:

f = conditional probability of incapacitation given whatever type of kill the host platform has suffered,

and using the personnel shaping parameter, ρ.

REFERENCES

1. SWP/MOUT Report to the Thirteenth Quadripartite Working Group on Army Operational Research, 1 March 1982.

2. Brewer, Gary D. and Shubik, Martin. The War Game: A Critique of Military Problem Solving, Harvard University Press, 1979.

3. BATTLE Documentation, Volume 1: Users Manual. US Army TRADOC Systems Analysis Activity, White Sands Missile Range, New Mexico 88002, November 1978.

4. Hartley, D. A., Heath, R. B. and Degelo, G. MAJ. ACABUG Concept Design: Weapons Effects, Urban Terrain and Force Organization Data Base. Technical Report TR 53-82, US Army TRADOC Systems Analysis Activity, White Sands Missile Range, New Mexico 88002, October 1982

5. Hartley, D. A. ACABUG Interim Software Manual. Technical Report No. 32-83, US Army TRADOC Systems Analysis Activity, White Sands Missile Range, New Mexico 88002 (To be published).

6. Schneider, G. M. and Bruell, S. C. Advanced Programming and Problem Solving, J. Wiley & Sons, 1981.

7. Clark, R. and Koehler, S. The UCSD Pascal Handbook: A Reference and Guidebook for Programmers, Prentice-Hall, 1982.

8. Allan, P. M. Review and Evaluation of Current Suppression Models with Proposal for Interim Model. Interim Note G-45, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland, May 1977.

9. Law, Averill M. and Kelton, W. David Simulation Modeling and Analysis, McGraw-Hill, 1982.

DISTRIBUTION LIST

	No. of Copies
HQDA	
(SAUS-OR)	1
(DAMO-FDD)	1
(DAMA-Z)	1
(DAMA-WSM-S)	1
(DAMO-FDQ)	1
COMMANDER	
HQ TRADOC (ATCD-A)	1
(ATCD-AC)	1
(ATCD-AS)	1
(ATCD-AR)	1
(ATCD-AU)	1
(ATCD-F)	1
(ATCD-M)	1
(ATCD-N)	1
(ATCD-P)	1
(ATTE-P)	1
(ATTG-A	1
USACAC (ATZL-CAS-OA)	1
(ATZL-SWN-T)	1
(ATZL-SWT-C)	1
(ATZL-SWP-TD)	1
(ATOR-CAM-T)	1
(CACDA-MID)	1
(CGSC-OTI)	1
(CGSC-BSD)	1
USAIC (ATSH-CD-CS-OR)	1
USASSC (ATZI-DCD-C)	1
USALOGC (ATZL-O)	1
(ATCL-C)	1
(ATCL-CFS)	1
USAARMC (ATSK-CD-SD)	1
USAFAC (ATSF-CA)	1
USAAVNC (ATZQ-D-CC)	1
USASIGCEN (ATZH-CDC)	1
USAMICOM (DRSMI-OT)	1
(DRSMI-RSL)	1
(ATSK-CC)	1
USAES (ATZA-CDE)	1
USAORDCENSCH (ATSL-CD-TCD)	1
USAICS (ATSI-DT-TI (MSD))	1
(ATSI-CD-CB)	1
(ATSI-DT-TI)	1
USAQMC (ATSM-CD)	1
USATNGSPTCEN (ATIC-ART-RL)	1
(ATTG-ATB-CT)	1

DISTRIBUTION LIST

	No. of Copies
COMMANDER	
USA CML & MP CEN & FM (ATZN-CM-CC)	1
(ATZN-MP-CCC)	1
(ATZN-MP-DE)	1
USAARDC (DRSMC-RAA (D))	1
COMMANDANT	
USAADASCH (ATSA-CDH)	1
MARINE CORPS WASH DC (CODE POG)	1
DIRECTOR	
USAMSAA (DRXSY-GA)	1
(DRXSY-GC)	1
USAHEL (DRXHG-CCD)	1
(DRXHE-CC)	1
USATRASANA (ATOR-TAB)	25
(ATOR-TSL)	6
CG	
MCDEC QUANTICO, VA (CODES D03, 091)	1

END

FILMED

9-85

DTIC